BEA WebLogic Integration™

Introducing B2B Integration

Release 2.1 Service Pack 1
Document Date: January 2002

Introducing B2B Integration

Part Number	Date	Software Version
N/A	January 2002	2.1 Service Pack 1

Contents

Index

About This Document

This document explains how to get started with B2B integration using the BEA WebLogic Integration™ software:

- Chapter 1, “Overview,” provides an overview of the B2B integration features of the WebLogic Integration software, a summary of the requirements of business-to-business (B2B) e-commerce, and a description of how you can meet the needs of your e-commerce business using WebLogic Integration.
- Chapter 2, “Getting Started with B2B Integration,” provides an architectural overview of the B2B integration component of WebLogic Integration, and a walkthrough of the tasks you need to complete to get your environment ready to exchange business messages with business partners.

What You Need to Know

This document is written for managers, system administrators, and programmers who are interested in understanding the architectural requirements for implementing and administering e-commerce collaborations based on WebLogic Integration. It is assumed that you have a working knowledge of the BEA WebLogic Server™ system, XML, Enterprise Java Beans, and Java programming.

e-docs Web Site

The WebLogic Integration product documentation is available on the BEA Systems, Inc. corporate Web site. From the BEA Home page, click Product Documentation or go directly to the Product Documentation page at `http://e-docs.bea.com`.

How to Print the 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 Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Integration documentation Home page, click the PDF Files button, and select the document you want to print.

If you do not have 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 2.1 Service Pack 1.

If you have any questions about this release of WebLogic Integration, or if you have problems installing and running the software, contact BEA Customer Support through BEA WebSUPPORT at http://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 file names 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. *Examples*: 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 Overview

Enterprises strive to integrate, automate, and streamline core internal and external business processes to improve their performance in today's dynamic business-to-business (B2B) electronic commerce (e-commerce) environment. These business processes drive a company's e-commerce interactions with their customers, partners, distributors, and suppliers; they can also streamline the company's internal business. Among the target business processes for B2B integration and automation are design and specification, manufacturing and testing, procurement, sales, fulfillment, customer service, and planning. To support these processes, B2B integration must support workflow processing, messaging and routing, and enterprise application integration.

BEA WebLogic Integration™ is an XML- and Java-based e-commerce platform that enables you to implement complex e-commerce systems on the Web. It helps you to quickly deploy e-commerce systems that link existing back-end applications, databases, customers, and partners into automatic and flexible electronic collaborations.

WebLogic Integration is a software framework and a set of services built on top of BEA WebLogic Server™. It builds upon the WebLogic Server foundation by adding a framework for business process management, application and data integration, and B2B integration. This document is an introduction to the latter component, that is, B2B integration: messaging, connectivity, and business protocols.

WebLogic Integration's B2B integration component is implemented entirely in Java and leverages the J2EE standard APIs. XML is used as a standard format for documents exchanged by business partners. WebLogic Integration supports HTTP because the World Wide Web is the ubiquitous communication medium for e-business.

WebLogic Integration simplifies the implementation and development of business-to-business trading networks, providing opportunities to integrate internal business processes with inter-enterprise business message exchange. A variety of deployment models are supported.

The following sections provide an introduction to B2B integration:

WebLogic Integration: Support for B2B Integration

WebLogic Integration provides an infrastructure platform for integrating business processes that can span multiple corporate departments, multiple enterprises across the Internet, or both. The WebLogic Integration platform supports the building of mission-critical, scalable, real-world e-commerce collaborations. Its features include:

- An open, nonproprietary architecture that leverages Java, J2EE, XML, HTTP, HTTPS, and other industry standards to allow rapid system and cross-platform integration, with low barriers to entry.
- Support for multiple business protocols, including XOCP, RosettaNet, ebXML and cXML, thus providing the ability to exchange business documents in a secure manner with a variety of trading partners, accommodating the complexity of heterogeneous platforms, different message structures, and different processes in various e-business environments.
- A variety of connectivity options, allowing rapid connectivity among many trading partners for both automated and semi-automated interactions. Connectivity and subsequent business collaborations between trading partner applications are possible entirely within an enterprise or a company, between trading partner applications across company firewalls and over the Internet, or in a combination of the two.
- Support for rapidly enabling trading partners through no-cost zeroweight client connectivity, using browser-based interfaces and file-sharing mechanisms.

- Peer-to-peer and mediated messaging models: peer-to-peer messaging allows direct messaging between partners, and mediated messaging supports tasks such as message routing, message content filtering, and value-added services.
- Tools and processes for the effective management of dynamic and diverse trading partner relationships.
- Robust support for secure, high-volume business transaction levels based on BEA's award-winning WebLogic Server™ technology. BEA's proven, high-availability, 24x7-managed application server technology, with dynamic load-balancing, multithreading, and failover without processing interruption, has delivered industry-leading results in thousands of the world's most demanding application environments.
- Reliable, role-based XML messaging that supports enhanced send and receive capabilities, including support for large messages.
- Conversation coordination to manage the execution and interaction of trading partner applications within a conversation and to manage conversation life cycles.
- Business process management workflow automation tool, providing a flexible and dynamic process-based approach to integrate public and private business processes.
- An SSL-based secure platform for conducting collaborations that supports digital signatures, digital receipts, nonrepudiation, and mutual (two-way) authentication using digital certificates among trading partners.
- A data repository and a set of design and configuration tools to define and manage the metadata and conversation definitions required for B2B integration.

WebLogic Integration also offers a number of features inherited from WebLogic Server, including:

- Ability to build custom portals by leveraging BEA WebLogic Personalization Server™ components
- Portal backbone
- Controlled and secure Web access to existing business data and applications
- Support for existing applications based on CORBA, EJB, BEA Tuxedo®, and COM+

Meeting the Requirements of Your E-Business

WebLogic Integration manages enterprise-to-enterprise collaborations, allowing heterogeneous enterprises to interact in diverse business transactions, which can be complex and long-running. The following sections describe requirements for the framework to build such a B2B e-commerce environment, and they explain how WebLogic Integration meets these requirements:

- Connecting Trading Partners
- Defining Conversations and Roles
- Managing Business Processes
- Supporting Business Protocols
- Ensuring the Security of Transactions
- Defining Collaboration Agreements
- Managing Conversations
- Managing Systems and Applications

Connecting Trading Partners

A trading partner joins one or more other trading partners to form an e-commerce community with a specific business purpose. Business partners in an e-commerce community can range in size from large enterprises to small divisions within an enterprise. One of the basic building blocks of B2B e-commerce is the trading partner, specifically, the trading partner applications that form the nodes in system-to-system interactions among business partners. An e-commerce community formed by a group of trading partners can:

- Exist entirely within a company, spanning multiple corporate departments (the business purpose for such a community might be inventory management, for example)

- Span multiple companies across firewalls and over the Internet (the business purpose might be supply chain management or multistep purchasing interactions, for example)
- Include trading partners both within a company and in other companies (one or more of the trading partners within a company communicates with trading partners in other companies across the Internet)

A trading partner must have a special identity that defines where it fits with the business purpose of the e-community. In the WebLogic Integration environment, a trading partner refers specifically to an entity that has an agreement with another entity to participate in a specific business exchange, or conversation, in a specific role that is defined for the conversation.

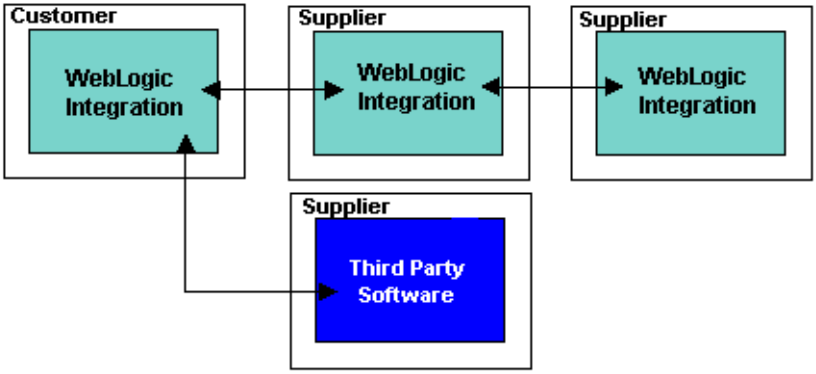
To meet the requirements of today's diverse B2B e-commerce activities, an enterprise must be able to use a variety of connectivity options. Such flexibility is necessary if a company wants to participate in business transactions with a large set of trading partners with diverse processes and protocols.

To that end, a WebLogic Integration trading partner application can be configured to communicate directly with other trading partners in a peer-to-peer mode, or through an intermediary in the hub-and-spoke mode, or both. These different configuration modes allow for either direct or mediated messaging among trading partners. An intermediary in the message flow can perform tasks such as routing and filtering of messages, or it can provide services to the trading partners in the conversation. For more details about modeling your B2B integration configurations, see "Configuration Models" on page 2-1.

Some business partners may have modest back-end integration requirements, or may need to participate in collaborative processes without installing the WebLogic Integration software. WebLogic Integration supports zeroweight clients to give small and medium-size enterprises, or enterprises with little or no back-end integration requirements, a simple integration path through which they can participate in e-business communities. Such enterprises can use a Web browser or a file-sharing client to communicate with business partners who deploy WebLogic Integration. The instance of WebLogic Integration to which they connect acts as a server for their needs. For details about setting up and configuring trading partner lightweight clients, see *Running the B2B Integration Samples*.

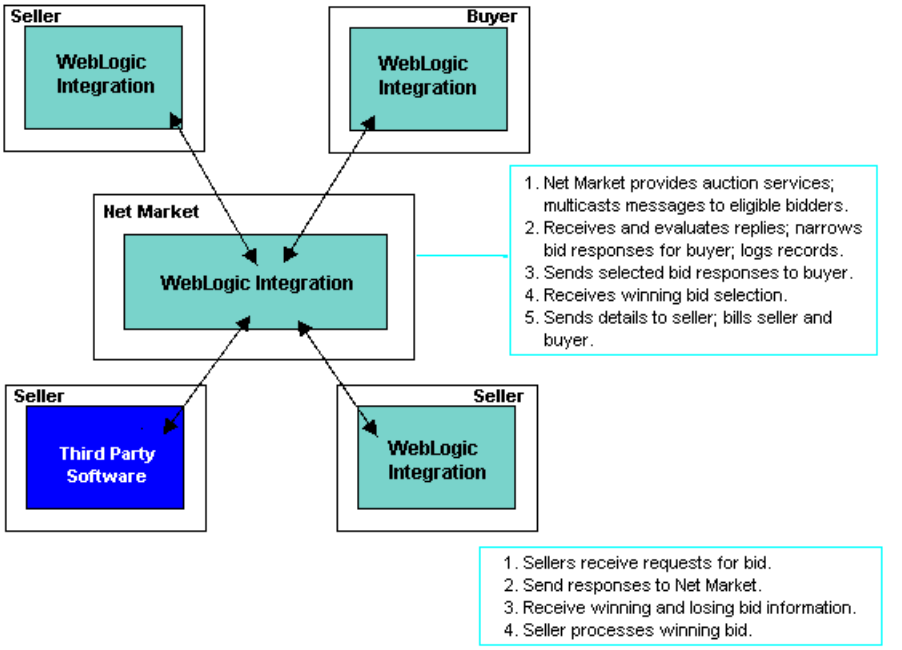
The following figure illustrates a simple scenario in which WebLogic Integration is deployed in peer-to-peer relationships between customers and suppliers in a value chain. A customer or supplier trading partner can support multiple peer-to-peer business partnerships with other trading partners.

Figure 1-1 Peer-to-Peer Messaging Among Trading Partners



The following figure illustrates a scenario in which WebLogic Integration is configured in hub-and-spoke mode. The Net Market trading partner, as the hub, provides intermediary services to several trading partners. For example, this might be an auction service, in which the hub is the auction broker.

Figure 1-2 Mediated Messaging in a Hub-and-Spoke Configuration



Defining Conversations and Roles

When trading partners join other trading partners to form an e-community with a specific business purpose, they participate in a *conversation*. A conversation:

- Is a series of business messages exchanged between trading partners
- May be complex and long-running, or short-lived
- Has a unique conversation name

The business messages that can be exchanged between participants in the conversation are determined by the roles the trading partners play in the conversation. The roles and other details of a conversation are specified in a conversation *definition* using the WebLogic Integration B2B Console. A *conversation* is an active instance of a *conversation definition.*

A conversation definition:

- Has a unique name and version.
- Defines two or more roles to be used by trading partners in a conversation. The kinds of messages trading partners may send and receive are dictated by their roles in the conversation.
- Is linked to a business protocol.
- Typically references a business process management collaborative workflow template for each role. The message choreography for a conversation is described in these collaborative workflow templates.

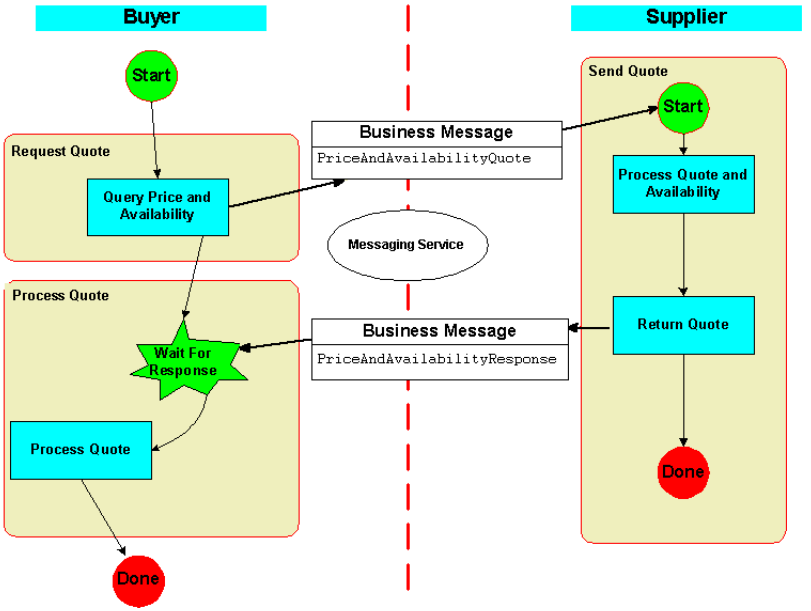
  Note: Using workflow templates is the recommended approach to composing business messages and choreographing their exchange in conversations. Alternatively, you can write Java messaging applications that use the WebLogic Integration Messaging API or the cXML API. When you use such messaging applications, conversation definitions do not reference a workflow template.

For details about writing messaging applications using the Messaging API and the cXML API, see *Programming Messaging Applications for B2B Integration* and *Implementing cXML for B2B Integration*, respectively.

When you use the WebLogic Integration Studio tool to compose business messages and manage their exchange in a conversation, each trading partner that participates in the conversation in a given role must implement the collaborative workflow required for its role. Collaborative workflows encapsulate the processes required to handle the right business messages at the right time for a given trading partner's role in a conversation.

For example, the following figure represents a simple conversation with two participating roles, buyer and supplier, and hypothetical workflows for the two roles.

Figure 1-3 Collaborative Workflows in a Query Price and Availability Conversation



In this figure, note the following:

- The business messages—Two business messages, `PriceAndAvailabilityQuote` and `PriceAndAvailabilityResponse`, are exchanged between trading partner applications.
- The buyer and supplier roles—The implication of being in a role in a given conversation is that you send and receive only the business messages defined for your role. For example, the buyer:
  - Starts the conversation
  - Sends the business message `PriceAndAvailabilityQuote`
  - Receives the business message `PriceAndAvailabilityResponse` and processes it

By contrast, the supplier:

- Receives and processes the business message `PriceAndAvailabilityQuote`
- Sends the business message `PriceAndAvailabilityResponse`

- The collaborative workflows—Each role has its own set of tasks required to send and receive the right business messages at the right times. A trading partner implements a collaborative workflow for a particular role in a conversation.

Managing Business Processes

Using workflows is the recommended approach to composing business messages and choreographing their exchange in conversations. Alternatively, you can write Java messaging applications that use the WebLogic Integration Messaging API or the cXML API. This section discusses the approaches to developing and managing messaging applications for B2B integration.

Workflows are business processes. Business processes can span multiple applications, corporate departments, and business partners (trading partners) behind a firewall and over the Internet. An enterprise's business processes can be divided into two broad categories: public and private.

Public and Private Business Processes

Business processes can be designed as public or private processes.

Public processes are *interface* processes. Their definitions and designs are known, understood, and agreed upon by the organizations using them, and may be customized or standardized across an industry or industry segment, as in the case of RosettaNet Partner Interface Processes (PIPs). They are part of a formal contract between trading partners that specifies the content and semantics of message interchanges. These processes can be implemented in different ways by different trading partners.

In the context of B2B integration, when collaborative workflows are intended to be reused in multiple conversations with different trading partners, the workflows should be designed as public processes.

Participants in a conversation can also implement private, noncollaborative workflows, which can integrate their back-end processing. Private processes are the business processes conducted within an organization. Their definitions and designs are specific to that organization and are not visible outside it. Within trading partner enterprises, private processes interface with public processes and with back-end business systems. In the context of public processes, private processes can be thought of as subworkflows or subprocesses that implement tasks that are part of the public workflow. For example, a trading partner may implement a private workflow that works in conjunction with a collaborative workflow and that implements the processes that occur locally to a trading partner, but that are not necessarily dictated by the conversation definition.

Business Process Management (BPM)

WebLogic Integration provides a business process management (BPM) component. It automates and integrates a business process by managing the sequence of activities in the process and invoking the appropriate resources required by the activities or steps in the process. The components of business process management include a GUI workflow design tool (the WebLogic Integration Studio), a GUI monitoring tool (the WebLogic Integration Worklist), and the process engine that monitors and controls workflows.

In the WebLogic Integration environment, a *collaborative workflow* is a workflow that implements a role in a conversation definition for a trading partner. The message choreography for a B2B conversation is defined by collaborative workflow templates: one template is defined for each role in the conversation definition, as described in “Defining Conversations and Roles” on page 1-7.

A B2B integration BPM plug-in extends the already powerful Studio design tool with functionality that allows you to create collaborative workflows for B2B integration. Using the plug-in functionality, you can compose and extract the contents of business messages, specify the message delivery Quality of Service (QoS), handle message tokens, and so on.

See *Creating Workflows for B2B Integration* for details about how to use the WebLogic Integration Studio to design collaborative workflows.

In summary, WebLogic Integration provides the following business process management functionality:

- A GUI design tool for creating workflows
- Functionality in the design tool for associating the workflows with a B2B conversation definition
- A run-time process engine for executing workflows
- Integration with back-end applications via the use of workflow actions
- Worklist: A utility for monitoring business processes and running workflows interactively
- Process monitoring capabilities

Using the Messaging API

As an alternative to using the WebLogic Integration Studio and other business process management capabilities, a trading partner can implement Java-based XOCP messaging applications based on the WebLogic Integration Messaging API. See *Programming Messaging Applications for B2B Integration* for information about using the Messaging API.

The Messaging API was available with WebLogic Integration Release 2.0. A messaging API (called the C-Enabler API) was also available with the WebLogic Collaborate product. See *Migrating to BEA WebLogic Integration Release 2.1* for details about migrating your C-Enabler and your Messaging API applications to WebLogic Integration Release 2.1.

Using the cXML API

WebLogic Integration B2B supports multiple business protocols for sending and receiving messages (see “Supporting Business Protocols” on page 1-13). For the cXML (Commerce eXtensible Markup Language) protocol, WebLogic Integration B2B provides a cXML API. The cXML API provides classes that allow the sending and receiving of cXML messages, the creation and manipulation of cXML documents, the mapping of a collaboration agreement to a message, and so on. For details, see *Implementing cXML for B2B Integration*.

Supporting Business Protocols

A business message is the basic unit of communication among trading partners and is exchanged as part of a conversation. A business message contains one or more XML business documents, one or more attachments, or a combination of both. The contents and format of a business message depend on the business protocol chosen for the conversation (see “Business Messages” on page 1-18 for details).

A business protocol is associated with a business process, which governs the exchange of business information between trading partners. It specifies the structure of business messages, how to process the messages, and how to route them to the appropriate recipients. A business protocol may also specify characteristics of messages related to persistence and reliability. You bind a business protocol to a conversation definition and a delivery channel for a trading partner. A business protocol is bound indirectly to a collaboration agreement through both the associated conversation definition and associated trading partner's delivery channel (see “Defining Collaboration Agreements” on page 1-25).

WebLogic Integration supports the following business protocols:

- XOCP (eXtensible Open Collaboration Protocol)
- RosettaNet
  - RosettaNet Implementation Framework (RNIF) 1.1
  - RosettaNet Implementation Framework (RNIF) 2.0
- cXML (Commerce eXtensible Markup Language)
  - cXML 1.1
  - cXML 1.2
- ebXML

By providing the ability to send and receive messages according to these standard protocols, WebLogic Integration gives an enterprise a great deal of flexibility and opportunity in organizing its B2B e-commerce by reducing the need for trading partners to standardize on any single protocol.

You can also customize and extend the supported business protocols beyond their out-of-the-box functionality by using logic plug-ins. Logic plug-ins are Java classes that can intercept and process business messages at run time. WebLogic Integration

provides system logic plug-ins, which you can supplement by writing custom logic plug-ins. The system logic plug-ins for XOCP include XOCP Router and XOCP Filter. They are directly involved in the processing of message recipients based on Xpath expressions in the repository. Custom logic plug-ins can perform a wide range of services that are unrelated to routing or filtering, as well as routing and filtering operations. For example, a custom logic plug-in might be used, for billing purposes, to track the number of messages sent from each trading partner.

See *Administering B2B Integration* and *Online Help for the WebLogic Integration B2B Console* for details about administering both system and custom logic plug-ins. See *Programming Logic Plug-Ins for B2B Integration* if you want to write custom logic plug-ins.

XOCP

The eXtensible Open Collaboration Protocol (XOCP) is a BEA-specific business protocol. The XOCP business protocol supports the standards-based ebXML Transport, Routing, and Packaging (TRP) protocol. XOCP provides the following messaging characteristics:

- Message multicasting

  Message multicasting is the ability of the XOCP messaging service to multicast a message from one trading partner to many trading partners, within the constraints of an existing conversation.

  For example, consider the following scenario:

  WebLogic Integration is deployed in a hub-and-spoke configuration, where 10 trading partners, configured as spokes, communicate via an intermediary (that is, through a trading partner configured as a hub).

  The business purpose of this e-community is participation in a Query Price and Availability conversation that has two roles: buyer and supplier. One trading partner is in the role of the buyer, and nine are suppliers. The buyer trading partner can, through the intermediary, multicast the Query Price and Availability message to all the trading partners who have agreements to be suppliers in a Query Price and Availability conversation with the intermediary.

- Message payload definition independence

  XOCP provides the flexibility for you to specify both the vocabulary and business processes for your messages for a given conversation so that they are an exact fit for your business requirements.

- Conversation life cycle management

  E-commerce conversations can be complex and long-lived. XOCP is designed to manage long-lived conversations. When a conversation terminates, all trading partners who are participating in that conversation receive an end-of-conversation message.

- Qualities of Service (QoS) capabilities

  A variety of settings related to Quality of Service allow you to set and control the characteristics of XOCP messages being sent, such as:

  - Message durability—Specifies whether a durable message store is to be used to guarantee the delivery of messages in the case of network, machine, or software failures.

    Note: Message durability is controlled at the message level by this QoS setting only when you deploy the B2B engine in nonpersistent mode. When you deploy the B2B engine in persistent mode, all state records, including messages, are read from and written to a persistent storage database and are not cached in memory. Therefore, all messages are persisted. In other words, in persistent mode, message durability is controlled by the database-based state management mode rather than by the message durability QoS setting. For details about persistent and nonpersistent modes, see "Configuring Persistence and Recovery" in *Administering B2B Integration*.

  - Timeout—Specifies how long a trading partner application waits before terminating all processing related to the business message.
  - Retry attempts—Controls how many times a message should be resent in the presence of specific situations, such as timeouts, network failures, and so on.
  - Correlation ID—Sets an additional business message property that can be used to correlate messages in a conversation.

  WebLogic Integration allows you to establish Quality of Service settings on a per-conversation and a per-message basis.

RosettaNet

WebLogic Integration supports sending and receiving RosettaNet messages according to both RNIF 1.1 and RNIF 2.0. It also supports interoperability with other RosettaNet partners. In addition, WebLogic Integration collaborative workflows can participate in RosettaNet Partner Interface Processes (PIPs).

RosettaNet is a self-funded, nonprofit consortium of major companies (from the information technology, electronic component, and semiconductor manufacturing industries) working to create and implement industry-wide, open e-business process standards. These standards form a common e-business language, aligning processes between supply chain partners on a global basis. (For complete details about the RosettaNet organization, see http://www.rosettanet.org.) To support its mission, RosettaNet provides specifications for the RosettaNet Implementation Framework (RNIF), the Partner Interface Processes (PIPs), and business and technical dictionaries.

RosettaNet (http://www.rosettanet.org) defines its Partner Interface Processes (PIPs) as follows:

"RosettaNet PIPs are specialized system-to-system XML-based dialogs that define business processes between trading partners. Each PIP specification includes a business document with the vocabulary, and a business process with the choreography of the message dialog. PIPs fit into seven clusters, or groups of core business processes, that represent the backbone of the supply chain network. Each cluster is further subdivided into segments, which are cross-enterprise processes involving more than one type of supply chain partner. Within each segment are individual PIPs."

PIPs contain one or more Activities, and Activities, in turn, specify Actions. PIPs apply to the following core processes:

- Administration
- Partner, Product, and Service Review
- Product Introduction
- Order Management
- Inventory Management
- Marketing Information Management
- Service and Support
- Manufacturing

The RNIF provides exchange protocols for implementation of the PIPs. The RNIF specifies information exchange between trading partner servers using XML, covering transport, routing and packaging, security, signals, and trading partner agreements.

For details about RosettaNet support in WebLogic Integration, see *Implementing RosettaNet for B2B Integration*.

cXML

WebLogic Integration provides a cXML API so that trading partner servers can send and receive cXML messages according to the cXML standard. The cXML API provides classes that allow the sending and receiving of cXML messages, the creation and manipulation of cXML documents, the mapping of a collaboration agreement to a message, and so on.

The `http://www.cxml.org` Web site defines cXML as follows:

"cXML is a streamlined protocol intended for consistent communication of business documents between procurement applications, e-commerce hubs, and suppliers."

cXML transactions consist of documents that are simple text files with well defined format and content. Most types of cXML documents are analogous to hardcopy documents traditionally used in business. The main types of cXML documents are Catalogs, Punchouts, and Purchase Orders. cXML defines a common DTD for business messages.

The cXML protocol is designed to link buyers and suppliers in scenarios where buyers browse catalogs and submit purchase orders to suppliers. A buyer can browse a supplier's catalog directly; alternatively, an Ariba exchange (Ariba Commerce Server Network) mediates messages between buyer and supplier.

For details about cXML support in WebLogic Integration, see *Implementing cXML for B2B Integration*.

ebXML

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. In addition, WebLogic Integration supports the creation and execution of workflows that model ebXML business messages.

The ebXML Message Service Specification is a set of layered extensions to the base Simple Object Access Protocol (SOAP) and SOAP Messages with Attachments specifications. The ebXML Message Service provides security and reliability features that are not provided in the specifications for SOAP and SOAP Messages with Attachments.

A trading partner that deploys WebLogic Integration can use ebXML to interoperate with a trading partner that deploys WebLogic Integration – Business Connect. WebLogic Integration – Business Connect is a lightweight client that can be deployed in a few hours, enabling enterprises to enlist new business partners quickly by eliminating their barriers to entry.

For information about WebLogic Integration – Business Connect, see *Using WebLogic Integration – Business Connect*.

For information about ebXML support in WebLogic Integration, see *Implementing ebXML for B2B Integration*.

Business Messages

A business message is the basic unit of communication among trading partners. It is exchanged as part of a conversation. A business message contains one or more XML business documents, one or more attachments, or a combination of both.

The contents and format of a business message depend on the business protocol chosen for the conversation. In WebLogic Integration, the primary method used to prepare a business message to be sent to one or more trading partners is a collaborative workflow. (See “Business Process Management (BPM)” on page 1-11.)

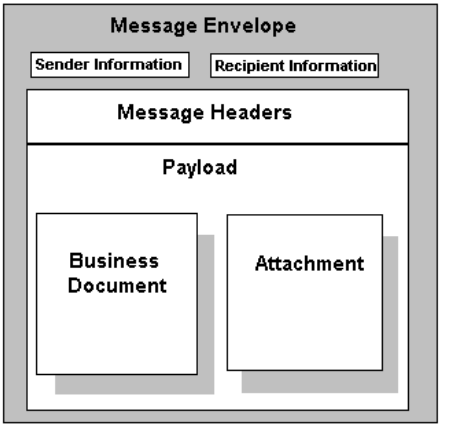
This section describes the basic structure of business messages exchanged in the B2B integration environment. See *Creating Workflows for B2B Integration* for details about using the WebLogic Integration Studio to compose and manipulate business messages.

XOCP Business Messages

The XOCP business protocol supports the open standard ebXML Transport, Routing, and Packaging (TRP) protocol (see `http://www.ebxml.org` for the “ebXML Message Service Specification”). The ebXML TRP protocol defines a wire format and protocol for a message service to support XML-based electronic messages.

The following figure represents the structure of a business message exchanged in a conversation based on the XOCP protocol.

Figure 1-4 XOCP Business Message



Note the following parts:

- Message envelope—Logical container for the XOCP business message that is added while the business message is in transit through the WebLogic Integration messaging service. The message envelope typically contains data related to the sender of the message and the recipients, and can contain other metadata. The envelope is visible in the messaging service only (see "Messaging Service" on page 1-28). The envelope is not part of the transport protocol.
- Message headers that include the following:
  - Transport, Security, and Qualities of Service (QoS) headers—Contain the data required to deliver a message, such as sender and recipient identities, content length and type, and status code.
  - Conversation header—Contains conversation information, sender information, message ID, creation timestamp, and other data. Message attributes can be set by the originating trading partner, or by logic plug-ins, such as the XOCP router or XOCP filter (see *Programming Logic Plug-Ins for B2B Integration*).
  - Routing header—Contains an expression that identifies a group of recipients for a multicast message.

- Payload—Includes one or more business documents, zero or more attachments, or a combination of both. A business document is an XML document, and an attachment can be any binary (nonXML) part of the message.

  The primary method used to prepare a business message payload to be sent to one or more trading partners is a collaborative workflow. That workflow includes a process that adds XOCP headers to an XOCP business message.

Note: The B2B engine supports sending encoded messages. An XOCP business message is encoded using the encoding specified in the WebLogic Integration repository for the recipient trading partner. See *Online Help for the WebLogic Integration B2B Console* and "Working with the Bulk Loader" in *Administering B2B Integration* for information about using the B2B Console and the Bulk Loader to configure trading partners.

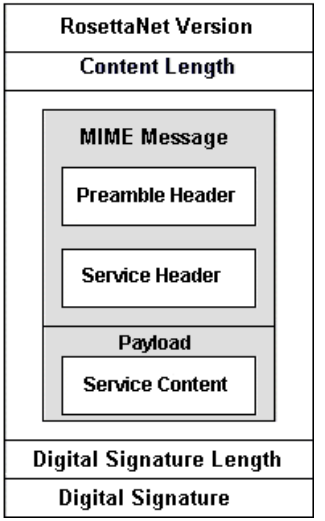
RosettaNet Business Messages

WebLogic Integration supports sending and receiving RosettaNet messages according to the RosettaNet Implementation Framework, versions 1.1 and 2.0. A business message exchanged in a conversation based on the RosettaNet 1.1 protocol is called a *RosettaNet Object* (RNO). The entity exchanged in a conversation based on the RosettaNet 2.0 protocol is called a *RosettaNet Business Message* (RBM).

A B2B integration BPM plug-in extends the functionality of the Studio with features that simplify the development of RosettaNet business messages and Partner Interface Protocols (PIPs).

The following figure represents the structure of a RosettaNet Object exchanged in a conversation based on the RosettaNet 1.1 business protocol.

Figure 1-5 RosettaNet Business Message



Note the following parts:

- Headers—A RosettaNet Object always contains a Preamble Header and a Service Header.
- Service Content—The payload part of the message. The Service Content contains either an action or a signal message.
- RosettaNet Version, Digital Signature, Content Length, and Digital Signature Length—These parts of the message are in binary format.
- Attachments are supported for RosettaNet Implementation Framework 1.1 and 2.0 as an optional part of the RosettaNet message. Attachments are not file type-specific, and may contain binary data. One or more attachments may be included in the payload part of the message.

The RosettaNet Implementation Framework 2.0 introduced the following notable differences in the composition of a RosettaNet Business Message (RBM):

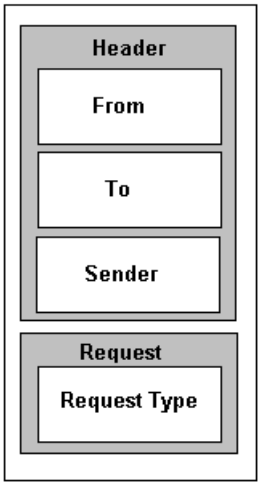
- The Service Header and Content in RosettaNet Business Messages can be encrypted. Using the WebLogic Integration B2B Console, you can configure the system to encrypt the Service Content, Service Header, and any attachments when a message is sent.
- In addition to the Service and Preamble Headers, the RosettaNet Implementation Framework 2.0 defines a Delivery Header.

Note: The B2B engine supports sending encoded messages. The B2B engine uses UTF-8 to encode all RosettaNet messages.

cXML Business Messages

WebLogic Integration supports sending and receiving cXML messages. cXML request and response transactions belong to one of four general types: Profile, PunchOut, Order, or Subscription transactions. The following figure represents the structure of an XML *request* business message exchanged in a conversation based on the cXML business protocol.

Figure 1-6 cXML Business Message



Note the following parts:

- Header containing *From*, *To*, and *Sender* fields:
  - From—Contains information about the originator of the request (the buyer).
  - To—Contains information about the destination of the request (the supplier).
  - Sender—Contains information about the entity that is relaying the request. Typically, this entity is an Ariba exchange (Ariba Commerce Server Network), which mediates messages between buyer and supplier.
- Request containing a *Request Type*

  A *Request Type* field contains information about the type of request contained in this message. For example, a request type may be PunchOutSetupRequest. Associated with each request type is one if three operation entities: create, edit, or inspect.

Note: The B2B engine supports sending encoded messages. The B2B engine uses UTF-8 to encode all cXML messages.

ebXML Business Messages

An ebXML business message contains one XML business document and one or more attachments. An ebXML message is a communication protocol-independent MIME/Multipart message envelope, referred to as a *message packag*e. All message packages are structured in compliance with the SOAP Messages with Attachments specification.

For information about ebXML business messages in WebLogic Integration, see "Using Workflows with ebXML" in *Implementing ebXML for B2B Integration*.

Ensuring the Security of Transactions

Reliable and secure communications are a crucial element in the B2B e-commerce environment. This is true whether e-business collaborations are between partners within an organization or between partners that span multiple organizations across firewalls and over the Internet.

WebLogic Integration B2B security is built on top of the security features provided by WebLogic Server; it provides advanced security support and services. WebLogic Integration B2B provides support for the following security features:

- An SSL-based secure platform for B2B e-commerce conversations
- Mutual, two-way authentication using digital certificates

  WebLogic Integration B2B provides support for certificate verification to authenticate the identities of trading partners. Using the B2B Console, administrators can configure a certificate verification provider implementation to verify a digital certificate submitted by a trading partner.

- Digital receipts

  A digital signature can be attached to a business document by the parties in a business collaboration. Digital signatures are supported at the application level; a trusted third party creates digital receipts.

- Nonrepudiation

  Nonrepudiation of origin and nonrepudiation of receipt is a legal requirement for critical business messages. Nonrepudiation of origin links the message received and the sender of the message, and nonrepudiation of receipt links the message processed and the recipient of the message. To support nonrepudiation, WebLogic Integration B2B provides the following services:

  - Digital signature—Used to digitally sign a business document before it is sent to the recipient.
  - Secure timestamp—Used to sequence the occurrence of events in the business transaction.
  - Audit log—Used to store digitally signed business messages with a secure timestamp. Audit logging is necessary for nonrepudiation.

- Data encryption

  WebLogic Integration B2B provides a data encryption service for business protocols that require this type of support.

See *Implementing Security with B2B Integration* for a comprehensive discussion about security, and details about configuring security services.

Defining Collaboration Agreements

A critical component of B2B e-commerce is the establishment and management of agreements among business partners. In the WebLogic Integration environment, these agreements take the form of collaboration agreements between trading partners. Using collaboration agreements, trading partners agree on the interactions between them, in particular, the conversations in which they participate, and each trading partner's message sending and receiving characteristics.

Parties in Collaboration Agreements

A conversation definition defines two or more roles to be used by trading partners in a conversation. A *party* in a collaboration agreement binds a role from the conversation definition to a trading partner. For example, consider a collaboration agreement based on the Query Price and Availability conversation described in "Defining Conversations and Roles" on page 1-7. There are two parties in this collaboration agreement: one buyer and one supplier (see Figure 1-7).

Delivery Channels in Collaboration Agreements

A trading partner's message sending and receiving characteristics are encapsulated in a *delivery channel*. There is generally one delivery channel per trading partner for each business protocol supported by the trading partner.

Note: Trading partners using the XOCP business protocol require two delivery channels: one hub and one spoke. For details about configuring delivery channels in hub-and-spoke mode, see "Hub-and-Spoke Configuration" on page 2-4.

An application for a trading partner communicates, through its delivery channel, with another trading partner's delivery channel. The interactions can be direct, that is, peer-to-peer between trading partners, or they can be indirect, that is, they can be conducted through an intermediary (routing proxy) delivery channel.

See "Configuration Models" on page 2-1 for details about peer-to-peer and hub-and-spoke configurations.

You configure and monitor the components of delivery channels using the B2B Console. A delivery channel includes the following information:

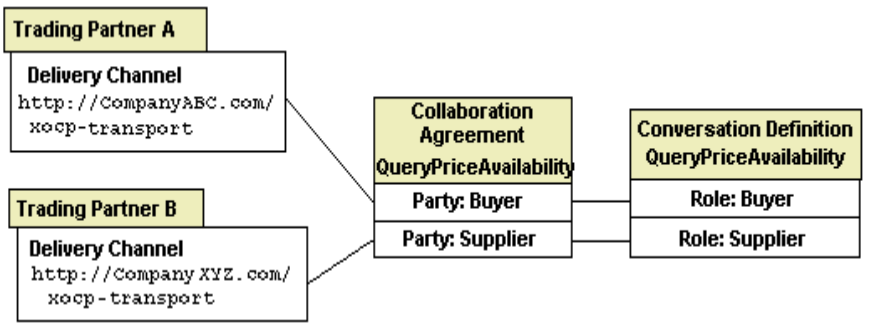
- Business protocol binding (XOCP, cXML, RosettaNet1.1, or RosettaNet2.0)
- Transport protocol (HTTP) and the target Uniform Resource Identifier (URI)
- Security options (digital signatures, digital certificates, and nonrepudiation)
- Reliable messaging options (timeout and retries)
- Message encoding (base64)

Collaboration Agreements in Peer-to-Peer Configurations

Take the Query Price and Availability conversation described in "Defining Conversations and Roles" on page 1-7 as an example. The delivery channels for the participating trading partners isolate the communication details between them.

The following figure illustrates the components of a collaboration agreement between two trading partners participating directly with each other in a business conversation. The trading partners are configured in a peer-to-peer configuration. In this case, the target trading partner for business messages is the other trading partner in the collaboration agreement.

Figure 1-7 Components of a Collaboration Agreement



The preceding figure illustrates how each trading partner's delivery channel and the conversation definition (which defines the roles in the conversation) relate to the collaboration agreement between the trading partners.

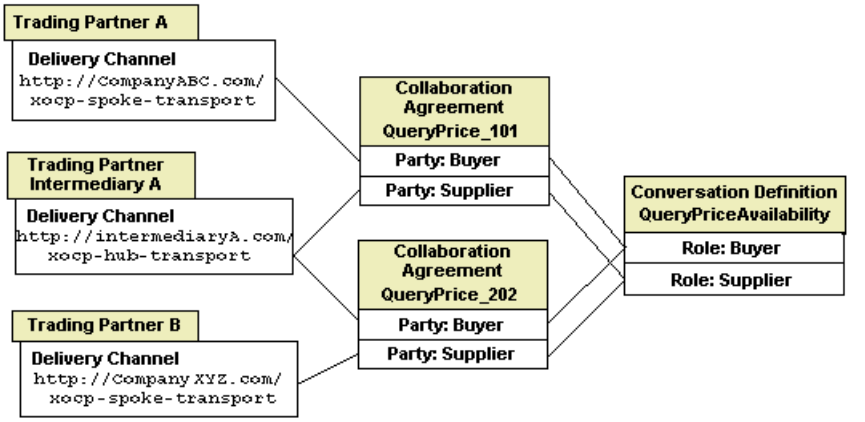
Collaboration Agreements in Hub-and-Spoke Configurations

When WebLogic Integration is configured so that trading partners participate in conversations through an intermediary (or routing proxy), collaboration agreements are created between each trading partner and the intermediary. In this case, trading partners identify the intermediary trading partner (that is, the routing proxy delivery channel) as the target for their business messages.

If, for example, the Query Price and Availability conversation between Trading Partner A and Trading Partner B goes through an intermediary trading partner (Intermediary A), two collaboration agreements are created.

The following figure illustrates the collaboration agreements created between Trading Partner A, Intermediary A, and Trading Partner B for a Query Price and Availability conversation in which Intermediary A is an intermediary trading partner.

Figure 1-8 Collaboration Agreements Between Trading Partners in a Hub-and-Spoke Configuration



Note the following in the preceding figure:

- The delivery channel for the trading partner (Intermediary A) in the role of routing proxy has collaboration agreements with both the supplier and buyer trading partners.
- In the collaboration agreement named QueryPrice_101, Trading Partner A is the buyer and Intermediary A is the supplier (in effect, the proxy supplier).

- In the collaboration agreement named QueryPrice_202, Trading Partner B is the supplier and Intermediary A is the buyer (in effect, the proxy buyer).

Managing Conversations

Critical to managing successful relationships between trading partners is the ability to provide robust services to ensure the integrity of business messages while they are being exchanged in various types of trading partner collaborations. WebLogic Integration provides a messaging service and a conversation coordination service, as described in the following sections.

Messaging Service

A flexible messaging service facilitates information transfer between trading partners. The messaging component relies on decoupled, deferred synchronous messaging capabilities to allow communication flexibility.

The messaging service offers the following features and characteristics:

- Support, in native mode, for multiple XML-based business protocols, including XOCP, RosettaNet 1.1, RosettaNet 2.0, and cXML
- Support for blocking and nonblocking message delivery
- Quality of Service (QoS) capabilities for XOCP-based messages, including the ability to establish durability, confirm receipt, track messages, message timeout, retry, and other settings for messages exchanged among trading partners
- Routing and filtering capabilities, controlling which trading partners receive messages
- Accommodation of the insertion of user-written code to customize the handling of incoming or outgoing messages or to perform messaging-related operations, thus extending the functionality of standard business protocols

Conversation Coordination Service

Business conversations can be complex and long running—some conversations may last several days. Conversations always have life cycles, which are explicitly demarcated by a beginning and an end. The WebLogic Integration software provides a conversation coordination service.

The conversation coordination service is business protocol-specific. Business protocols define their conversation termination protocols in different ways. In the case of XOCP-based conversations, the conversation coordinator does the following:

- Creates the conversation on behalf of the trading partner that initiates the conversation
- Enlists new trading partners in the conversation
- Removes trading partners from the conversation if they submit requests to leave the conversation
- Sends conversation termination notices to all conversation participants when the conversation is terminated
- Allows monitoring of the active conversations for a trading partner

Managing Systems and Applications

Workflow management is achieved through a process engine that controls the execution of local business processes and manages the integration of back-end enterprise applications with business processes. See “Managing Business Processes” on page 1-10 for more details.

Except for workflows, B2B integration components are configured and managed primarily through the B2B Console, which works together with a repository service.

This section describes management services that support B2B integration:

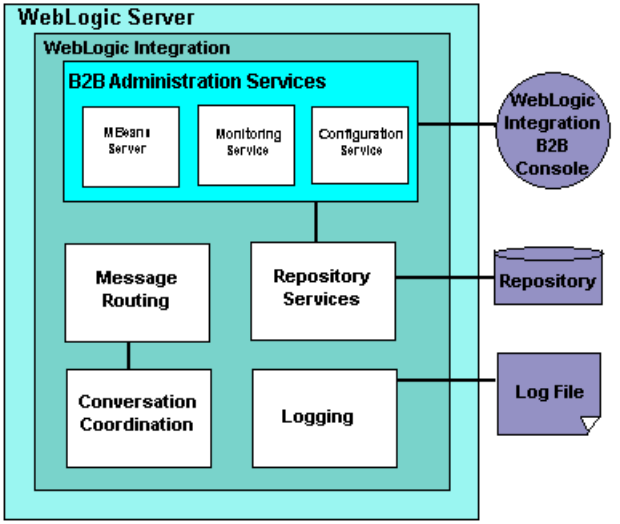
- Repository Service
- Administration Services
  - WebLogic Integration B2B Console
  - Bulk Loader

- Java Management Extensions Management Beans

- Logging Service

The following figure represents the WebLogic Integration services described here and in “Managing Conversations” on page 1-28.

Figure 1-9 WebLogic Integration B2B Services



Repository Service

The repository service stores data into the repository. This data is required by trading partners to engage in business-to-business collaborations, and includes:

- Trading partner-specific information
- Conversation definitions
- Collaboration agreements
- Business protocol definitions
- Document definition (DTD) names

- Document and configuration information
- Data used for message routing and filtering operations

In addition, the repository:

- Is accessible through the B2B Console for system administration, creation of collaboration agreements by business developers, and monitoring of trading partners, conversations, collaboration agreements, and so on.
- Supports the import and export of data using the B2B Console and a Bulk Loader utility.
- Supports Oracle, Microsoft SQL Server, Sybase, DB2, and Cloudscape as primary data stores. See *Installing BEA WebLogic Integration* and *BEA WebLogic Integration Release Notes* for information about database support.

For details about the repository and the Bulk Loader, see "Working with the Repository" and "Working with the Bulk Loader" in *Administering B2B Integration*.

Administration Services

The administration services support multiple system management functions, including configuring, administering, and monitoring trading partners, conversations, collaboration agreements, and more.

Through these services, an administrator can create, configure, and manage the B2B integration components of the system. Business developers can set up collaboration agreements and monitor system status. In addition, business partners can use the administration services to start and end trading partner sessions, leave or end conversations, and enable, disable, start, or shut down delivery channels.

WebLogic Integration B2B Console

Primary access to the administration services is through the Web-based B2B Console. All configuration information is stored in the WebLogic Integration repository, which is supported by a database management system. You can configure and monitor the following components:

- Trading partners
- Conversations

- Collaboration agreements
- Business protocols
- Logic plug-ins

To facilitate data and business process exchange among trading partners, the import and export functions of the B2B Console allow you to export repository data to an XML file, and import data from an XML file to your WebLogic Integration repository.

You can select the scope of the exported data from the repository such that the XML file you create contains either all the data in the repository, or only data related to one of the following entities: trading partners, conversation definitions, collaboration agreements, business protocol definitions, or logic plug-ins.

See *Administering B2B Integration* and *Online Help for the WebLogic Integration B2B Console* for complete details about using the B2B Console.

Bulk Loader

In addition to using the B2B Console to export and import data from and to your WebLogic Integration repository, you can use the Bulk Loader facility (from the command line) to export and import repository data. Like the import and export functions of the B2B Console, the Bulk Loader utility supports export of either all the data in the repository or only a subset of the data to the XML file. See "Using the Bulk Loader" in *Administering B2B Integration* for complete details about using the bulk loader.

Java Management Extensions Management Beans

For monitoring only, WebLogic Integration supports user-written applications that use Java Management Extensions (JMX) Management Beans (MBeans) to view data and statistics maintained by the administration services.

Trading partners can use the JMX MBeans through an MBean server (a repository for MBeans), which is included in the administration services package, to send and receive messages to and from the administration services.

See *Programming Management Applications for B2B Integration* for details about programming management applications.

Logging Service

The B2B integration component of WebLogic Integration provides a logging capability for error and information messages. All log messages are time-stamped, and can be sent to the WebLogic Server log, or to a separate log file. For details, see *Writing Messages to the B2B Integration Log*.

CHAPTER

2 Getting Started with B2B Integration

This section describes how the architecture of WebLogic Integration provides a robust framework supporting B2B integration: messaging, connectivity, business protocols, and integration with business process management and workflow automation. It includes the following topics:

- Configuration Models
- B2B Integration: A Walkthrough

Configuration Models

In order to participate in business transactions with a large set of trading partners who use diverse processes and protocols, an enterprise requires a variety of connectivity options.

To that end, a trading partner using WebLogic Integration can be configured to communicate in a number of different ways. Trading partner applications communicate through their delivery channels either directly, in peer-to-peer mode, or through an intermediary delivery channel, in hub-and-spoke mode, or both. WebLogic Integration can simultaneously support multiple peer-to-peer relationships, and serve as a routing hub, mediating messages among other trading partners that deploy the WebLogic Integration B2B engine. Multiple messaging models can be overlaid on these configurations.

The following sections describe configuration and messaging models for trading partners:

- Peer-to-Peer Configuration
- Hub-and-Spoke Configuration

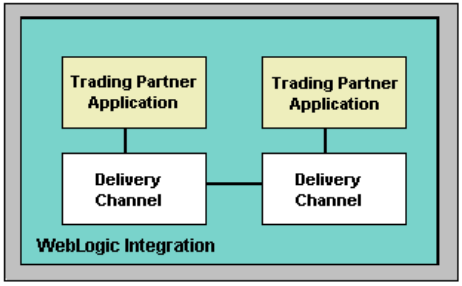
Peer-to-Peer Configuration

In a peer-to-peer configuration, applications for two trading partners communicate through their respective delivery channels. A trading partner's message sending and receiving characteristics are encapsulated in a delivery channel (see "Delivery Channels in Collaboration Agreements" on page 1-25).

Multiple trading partner applications can be implemented in the following configurations:

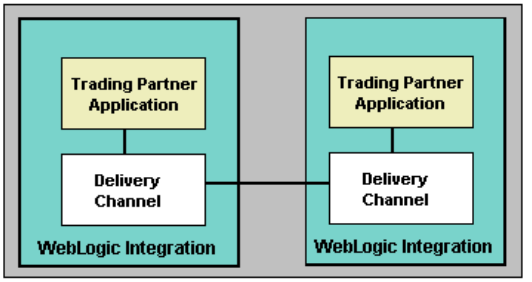
- On a single instance of WebLogic Integration within a company, as shown in the following figure.

Figure 2-1 Peer-to-Peer Configuration: Single Instance WebLogic Integration



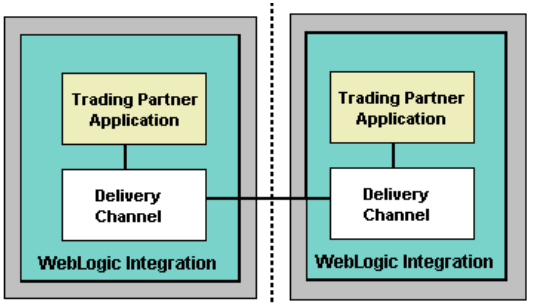
- On multiple instances of WebLogic Integration within a company or organization, as shown in the following figure.

Figure 2-2 Peer-to-Peer Configuration: Multiple Instances WebLogic Integration in a Single Company



- On multiple instances of WebLogic Integration across a firewall and over the Internet, as shown in the following figure.

Figure 2-3 Peer-to-Peer Configuration: Multiple Instances WebLogic Integration in Multiple Companies



Peer-to-peer configurations support the exchange of business messages using the RosettaNet, cXML, and ebXML business protocols.

See "Configuration Tasks" in *Administering B2B Integration* for details about specific configuration tasks.

Collaboration agreements between trading partners capture the information necessary for routing business messages among trading partners. In the case of peer-to-peer communication, the target trading partner for a business message is the other trading partner in the collaboration agreement (or the role of the other trading partner, if there are more than two roles defined for the conversation).

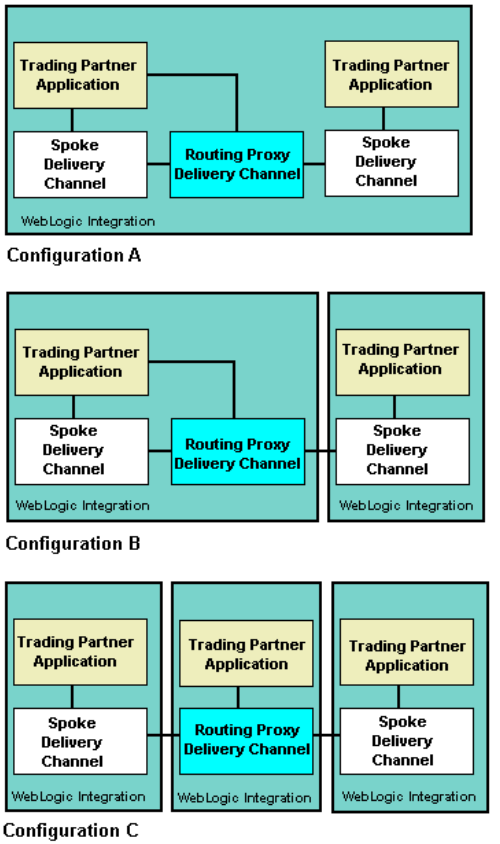
Hub-and-Spoke Configuration

In a hub-and-spoke configuration, a trading partner application communicates via an intermediary (routing proxy) delivery channel to another trading partner's delivery channel (spoke delivery channel). The following figure includes illustrations of possible hub-and-spoke configurations for the delivery channels, the trading partner applications, and the instances of WebLogic Integration that host the trading partner applications.

Caution: Configurations A and B in the following figure show a single trading partner configured with two delivery channels; one hub, one spoke. There is currently a limitation with this configuration. Do not configure the hub and spoke delivery channels on one trading partner. Instead set up two trading partners, each with one delivery channel. Configure a hub delivery channel for one trading partner, and a spoke delivery channel for the other.

See *Administering B2B Integration* for details about configuring trading partners and delivery channels.

Figure 2-4 Hub-and-Spoke Configuration Models



The hub-and-spoke configuration models in the preceding figure can be summarized as follows:

- Configuration A: Two trading partners are configured on a single instance of WebLogic Integration. Two delivery channels, one hub and one spoke, are configured for one trading partner. A single delivery channel, a spoke, is configured for the second trading partner.
- Configuration B: Two instances of WebLogic Integration are deployed; a single trading partner is configured for each instance. Two delivery channels, one hub and one spoke, are configured for one trading partner. A single delivery channel, a spoke, is configured for the second trading partner.

- Configuration C: Three instances of WebLogic Integration are deployed. A single trading partner is configured for each instance, and a single delivery channel is configured for each trading partner. The hub delivery channel, acts as the intermediary between the two spoke delivery channels.

Hub-and-spoke models support the exchange of business messages using the XOCP business protocol. XOCP business messages can be unicast or multicast among trading partners. Message multicasting is the ability of the XOCP messaging service to broadcast a message from one trading partner to many trading partners, within the constraints of an existing conversation.

Collaboration agreements between trading partners capture the information necessary for routing business messages among trading partners. For business messages to be exchanged between trading partners through an intermediary (hub), collaboration agreements are created between each trading partner and the intermediary. In contrast to the exchange of business messages between trading partners that communicate directly (when the target trading partner for a business message is the other trading partner in the collaboration agreement), trading partners using an intermediary identify the intermediary trading partner (that is, the routing proxy delivery channel) as the target for their business messages. The destination for the message is specified when the message is sent and, in the case of XOCP business messages, can be a single destination (a unicast message) or multiple destinations (a multicast message). Recipient trading partners can be represented in a message by an XPath expression.

See “Supporting Business Protocols” on page 1-13 for a discussion of the XOCP protocol and message multicasting.

Message Mediation Models

In the hub-and-spoke model, WebLogic Integration is installed at both the hub and spoke nodes, as described in the previous section. Messages can be mediated in any of the following ways:

- Routing—Destination trading partners are known to the message initiator. WebLogic Integration, configured as a hub, acts as a routing proxy and passes business messages to the recipients indicated by the sender.
- Interposed workflows—Destination trading partners are not known to the message initiator. There are collaboration agreements between trading partners and the intermediary, but not between the trading partners themselves. The intermediary intercepts the messages with a workflow that can do any type of processing, including complex filtering and routing, integration with back-end

systems, message validation, logging, and so on. This model is common in blind auctions, in which the hub is the auction broker.

- Parallel workflows—The message initiator has collaboration agreements with the hub, a trading partner collocated at the hub, and other destination trading partners routed by the hub. A message goes to all trading partners, including the collocated partner at the hub. On the hub, messages are directed to parallel workflows: one workflow passes messages on to their destination trading partners; another mediates messages by performing business processing.

  Note: When you define a conversation in the WebLogic Integration B2B Console, you can mark a workflow for a role in the conversation as a *proxy* workflow. The hub delivery channel uses this *proxy* flag to differentiate between collaboration agreements that apply to conversations with a role local to the hub, and collaboration agreements that require the hub to be in the role of routing proxy.

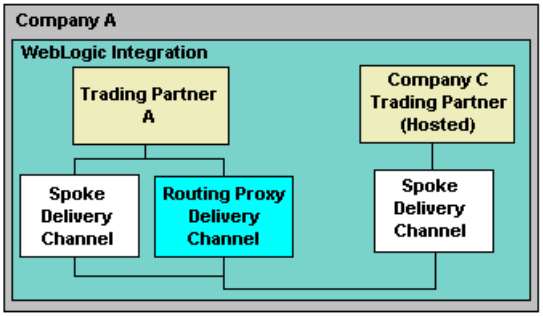
- Interceptors—Trading partners know about each other (either a routing model or a parallel workflows model is in place), but the hub uses logic plug-ins to intercept messages and do additional processing. Logic plug-ins are Java classes that can intercept and process business messages at run time. WebLogic Integration provides system logic plug-ins, which you can supplement by writing custom logic plug-ins. The system logic plug-ins include XOCP router, XOCP filter, and RosettaNet logic plug-ins, all of which are directly involved in the processing of message recipients based on XPath expressions in the repository. Custom logic plug-ins can perform a wide range of services that are unrelated to routing or filtering, as well as routing and filtering operations. Multiple logic plug-ins can be installed in chains. A chain of routing logic plug-ins can be invoked when messages are sent, and a chain of filtering plug-ins can be invoked when messages are received.

Trading Partner Zeroweight Clients

As well as making direct connections between trading partner applications, WebLogic Integration supports zeroweight clients to give small and medium-size trading partners, or trading partners with little or no back-end integration requirements, ways to connect and participate in e-business communities. Such trading partners communicate with WebLogic Integration using either a Web browser or a file-sharing client. Messages for these zeroweight clients are processed by the instance of WebLogic Integration to which they connect. In other words, message processing is hosted on behalf of the trading partner deploying the Web browser or file-sharing

client. The following figure represents an example configuration in which two trading partners are created on Company A's instance of WebLogic Integration, one of which hosts the Web browser or file-sharing client used by Company C.

Figure 2-5 Hosted Trading Partner Configuration



For details about configuring and deploying trading partner zeroweight clients, see "Trading Partner Zeroweight Client Sample" in *Running the B2B Integration Samples*.

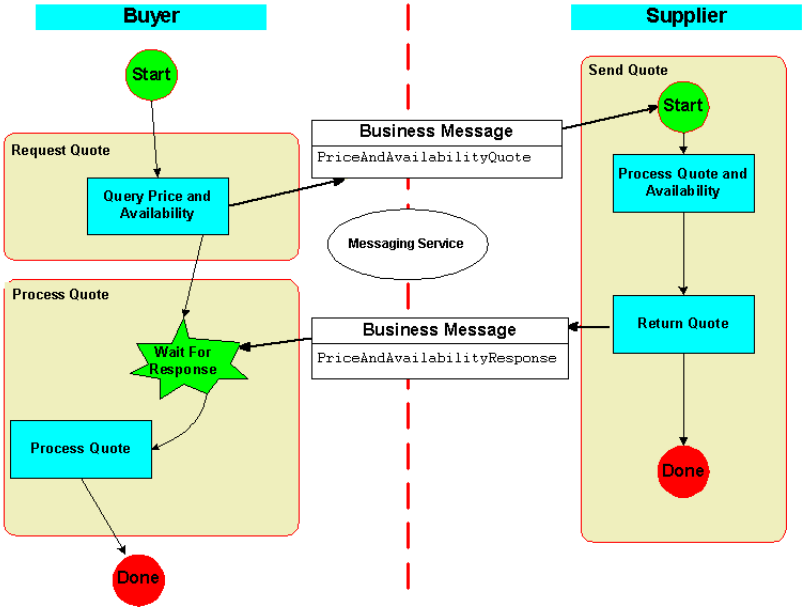
B2B Integration: A Walkthrough

Using a Query Price and Availability conversation as an example, this section describes:

- The approach for setting up the WebLogic Integration components to allow one trading partner to participate in a conversation with another trading partner, which shows how the components work together for B2B collaborations
- The exchange of business messages among trading partners participating in a conversation, which shows how B2B integration works

The Query Price and Availability conversation used in this example is described in "Defining Conversations and Roles" on page 1-7, from which the following figure, illustrating the conversation, is taken.

Figure 2-6 Workflows in a Query Price and Availability Conversation



For the purposes of this scenario, there are two business partners:

- Company ABC—the buyer
- Company XYZ—the supplier

In this scenario, Company ABC is responsible for preparing the WebLogic Integration environment to participate in this example business collaboration. Before going into production with Company XYZ, Company ABC tests the components of the collaboration in its own WebLogic Integration environment.

The following sections describe how Company ABC sets up two trading partners in its WebLogic Integration environment to participate in this business collaboration. Next, the subsequent exchange of business messages among trading partners is described. Finally, the steps both business partners take to exchange the data required before they can participate in the Query Price and Availability conversation is described:

- Create Conversation Definitions
- Create Workflow Templates
- Create Trading Partners and Delivery Channels
- Create Collaboration Agreements
- Send and Receive Business Messages
- Export and Import Repository Information
- Start Business Collaboration

A detailed discussion of how to use the WebLogic Integration B2B Console or the WebLogic Integration Studio for configuration is beyond the scope of this section. For complete configuration details, see *Administering B2B Integration* and *Online Help for the WebLogic Integration B2B Console*.

Create Conversation Definitions

To specify the roles and other details of a conversation, you define a conversation using the B2B Console. The entity you define is known as a *conversation definition*. A *conversation* among trading partners is an active instance of a *conversation definition*.

In this example, an administrator for Company ABC uses the B2B Console to create a conversation definition by performing the following steps:

- Name the conversation definition. (In this example, the conversation definition is Query Price and Availability.)
- Define the roles in the Query Price and Availability conversation: buyer and supplier.
- Reference XOCP as the business protocol to be used in this conversation. (The other choices are cXML1.1, cXML1.2, RosettaNet 1.1, and RosettaNet 2.0.)

- Attach a collaborative workflow template name to each role.

  Note: The workflow template names you attach during this step are those of the workflow templates you must define in the WebLogic Integration Studio. For details, see "Create Workflow Templates" on page 2-11.

  The following templates are attached to the roles in this example Query Price and Availability conversation.

Role in Conversation	Collaborative Workflow Template
Buyer	`QueryPriceAvailability_Buyer`
Supplier	`QueryPriceAvailability_Supplier`

Create Workflow Templates

Workflows that implement the roles for trading partners in a conversation definition are referred to as *collaborative workflows.*

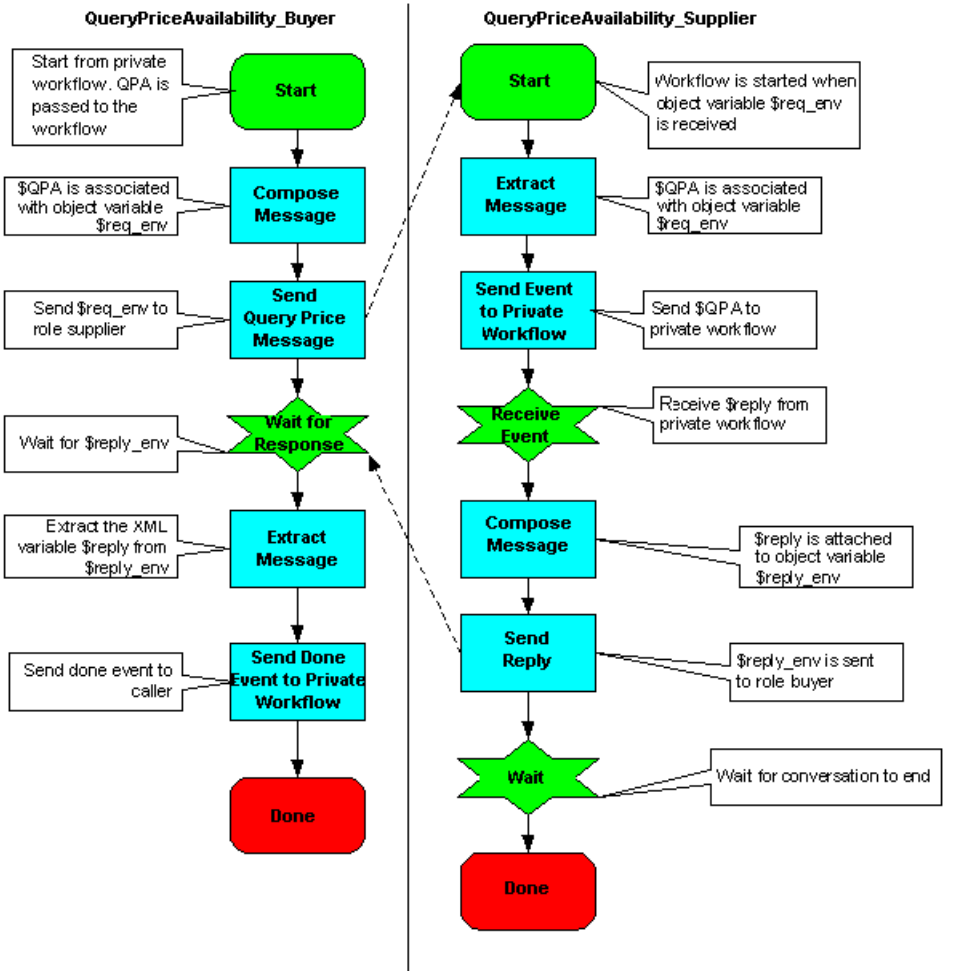
To design or edit workflow templates that implement conversation roles in a conversation, you use the WebLogic Integration Studio, specifically, the B2B integration BPM plug-in, which extends the functionality of the Studio to support collaborative workflows. The Studio allows you to assign properties to the workflows. These properties make the workflows usable in the B2B integration environment.

In this example, an administrator from Company ABC uses the WebLogic Integration Studio to create two collaborative workflow templates (one for each role in the conversation): `QueryPriceAvailability_Buyer` and `QueryPriceAvailability_Supplier`.

A trading partner may also implement private workflows that work in conjunction with a collaborative workflow and that implement local processes for a trading partner. Such processes are not necessarily dictated by the conversation definition (see "Create Conversation Definitions" on page 2-10). For example, in the case where a trading partner starts a conversation, that trading partner's private workflow can start the collaborative workflow to initiate the conversation.

The following figure displays example collaborative workflows for the buyer and supplier roles in the Query Price and Availability conversation.

Figure 2-7 Collaborative Workflows for Buyer and Supplier Roles



To prepare collaborative workflows to exchange business messages, you specify conversation properties for the collaborative workflow templates. These conversation properties are specified in the WebLogic Integration Studio using workflow template definitions.

To specify conversation properties for this example, you must complete the following tasks:

- Identify the `QueryPriceAvailability_Buyer` collaborative workflow template as the *Exchange Initiator.* (This specifies that the `QueryPriceAvailability_Buyer` workflow starts the conversation.)
- Link the `QueryPriceAvailability_Buyer` template with the buyer role in the Query Price and Availability conversation.
- Link the `QueryPriceAvailability_Supplier` template with the supplier role in the Query Price and Availability conversation.
- Set run-time properties for workflow templates. Such properties include:
  - General properties, such as template effective and expiry dates
  - Quality of Service (QoS) properties, such as number of retries, timeouts, and confirmation of delivery options

  Note: At run time, the QoS properties assigned to workflows take precedence over QoS properties that you define when you set up delivery channels for trading partners in the B2B Console.

See *Creating Workflows for B2B Integration* for details about how to use the WebLogic Integration Studio to design collaborative workflow templates.

Create Trading Partners and Delivery Channels

Enterprises that want to participate in a conversation must first configure their environments. This task includes creating a trading partner, and defining trading partner-specific information and one or more delivery channels. A delivery channel describes a trading partner's message sending and receiving characteristics, including the business protocol to be used in the conversation, a transport protocol, and security parameters.

Create Trading Partners for Company ABC

Using the B2B Console, an administrator for Company ABC creates two trading partners:

- Trading Partner A in the buyer role for the conversation
- Supplier-Test in the supplier role for the conversation

  Supplier-Test is used for testing the conversation and workflows before bringing the system up for production with a trading partner (in the role of supplier) from Company XYZ.

The information to create each trading partner includes:

- Name: Trading Partner A and Supplier-Test
- Type: Local
- Contact information: Address, email, phone, fax, and so on
- Party ID: A unique name that identifies the trading partner in a collaboration agreement (for example, DUNS A12345678)
- Delivery channels for Trading Partner A

  Trading Partner A requires two delivery channels: one spoke and one routing proxy. They are defined as follows:

  - Name: Spoke Delivery Channel—contains the following elements:

    a. Document Exchange: Defines the business protocol (XOCP) and run-time parameters

    b. Transport: Defines the transport protocol (HTTP), end point URI (for example, `http://CompanyABC.com/xocp-spoke-transport`), and security parameters

  - Name: Routing Proxy Delivery Channel—contains the following elements:

    a. Document Exchange: Defines the business protocol (XOCP) and run-time parameters

    b. Transport: Defines the transport protocol (HTTP), end point URI (for example, `http://CompanyABC.com/xocp-hub-transport`), and security parameters

- Delivery channel for Supplier-Test

  The Supplier-Test trading partner requires one spoke delivery channel. The delivery channel definition contains the following information:

  - Document Exchange—defines the business protocol for this trading partner (XOCP) and run-time parameters
  - Transport—defines the transport protocol (HTTP), end point URI (for example, `http://CompanyABC.com/xocp-spoke-transport`), and security parameters

Create Trading Partner for Company XYZ

After Company XYZ installs WebLogic Integration, an administrator uses the B2B Console to create a trading partner: *Trading Partner Z* in the supplier role.

The information to create Trading Partner Z includes:

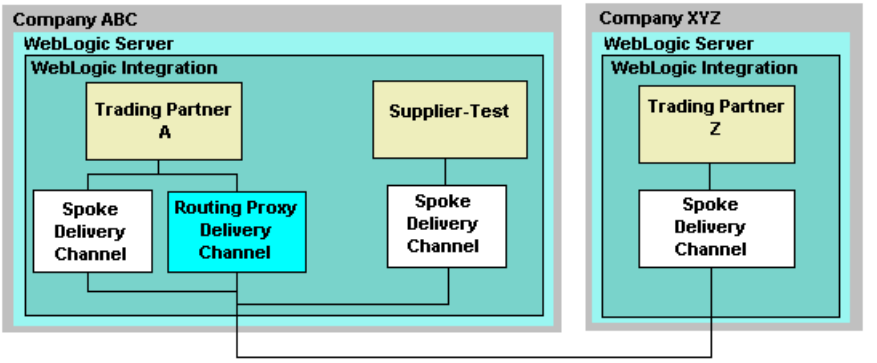
- Name: Trading Partner Z
- Type: Local
- Contact information: Address, email, phone, fax, and so on
- Party ID: A unique name that identifies the trading partner in a collaboration agreement (for example, DUNS Z87654321)
- Delivery channel

  Trading Partner Z requires one spoke delivery channel. The delivery channel definition contains the following information:

  - Document Exchange—defines the business protocol for this trading partner (XOCP) and run-time parameters
  - Transport—defines the transport protocol (HTTP), end point URI (for example, `http://CompanyXYZ.com/xocp-spoke-transport`), and security parameters

The following figure illustrates the configuration of the three trading partners and their delivery channels as defined for this scenario.

Figure 2-8 Configuration of Trading Partners in Example Conversation



Company ABC can use the Supplier-Test trading partner to test the collaboration agreement for the Query Price and Availability conversation before connecting with Company XYZ's trading partner, who will be the supplier in the real-life scenario.

Create Collaboration Agreements

A collaboration agreement binds the trading partners' delivery channels to the conversation definition. It is through this collaboration agreement that trading partners agree on the interactions between them, in particular, the conversation in which they participate and each trading partner's message sending and receiving characteristics.

An administrator uses the B2B Console to create a collaboration agreement, with a specific name and version, in its repository. A collaboration agreement defines a *party* for each trading partner participating in the agreement, and includes the following information:

- A unique party identifier for each party to the collaboration agreement.
- The delivery channel information for each trading partner, as described in "Create Trading Partners and Delivery Channels" on page 2-13.

The delivery channels define the business protocol for the collaboration, the run-time parameters for each trading partner, the transport protocol, the end point (URI) for each trading partner's transport, and security parameters.

- The conversation definition, as described in "Create Conversation Definitions" on page 2-10.

  The conversation definition names the business collaboration (in this case, Query Price and Availability), and attaches the roles (for example, buyer and supplier) to collaborative workflow templates.

Returning to our walkthrough example:

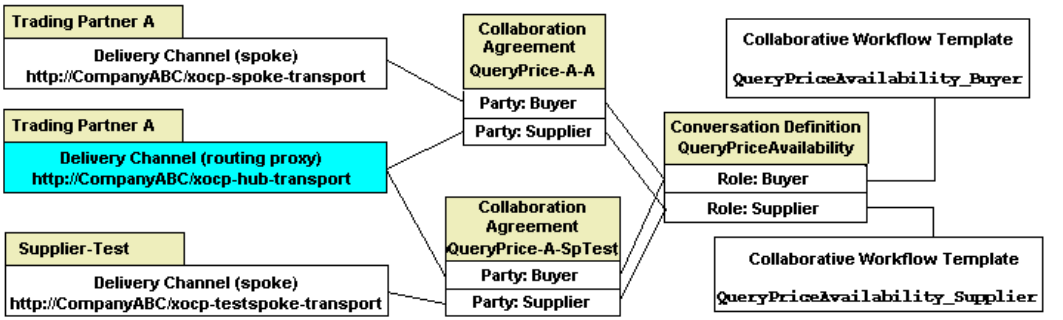
Company ABC creates two collaboration agreements as follows:

- QueryPrice-A-A—The parties are the Trading Partner A routing proxy delivery channel and the Trading Partner A spoke delivery channel.
- QueryPrice-A-SpTest—The parties are the Trading Partner A routing proxy delivery channel and the Supplier-Test spoke delivery channel.

Note that each trading partner has a collaboration agreement with the routing-proxy delivery channel. The routing proxy acts as an intermediary; that is, it acts as the proxy supplier in its collaboration agreement with Trading Partner A, and as a proxy buyer in its collaboration agreement with Supplier-Test.

The collaboration agreements bind the delivery channels to the conversation definition, and they reference the collaborative workflow templates that are associated with the roles in the conversation definition. The following figure represents the two collaboration agreements.

Figure 2-9 Components in the Collaboration Agreements



Notice the following in the preceding figure:

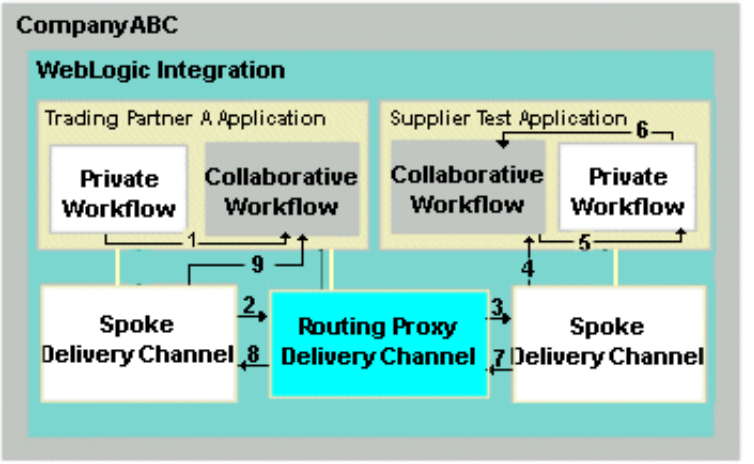
- The routing-proxy delivery channel is a party in both collaboration agreements. In the `QueryPrice-A-A` agreement, the routing-proxy delivery channel is in the role of (proxy) supplier, and in the `QueryPrice-A-SpTest` agreement, it is in the role of (proxy) buyer.
- The parties in the collaboration agreements are associated with the roles in the conversation definition.
- The conversation definition associates the collaborative workflow template for each role in the conversation with the parties in the collaboration agreement.

Send and Receive Business Messages

After the workflow templates, the conversation definition, the trading partners, the delivery channels, and the collaboration agreements have been created, Company ABC tests its collaboration agreements and conversation using Trading Partner A in the buyer role, and Supplier-Test in the supplier role for the Query Price and Availability conversation.

This section describes how business messages are sent and received during the Query Price and Availability conversation.

Figure 2-10 Sending and Receiving Business Messages



The preceding figure shows the processing of business messages through the trading partner applications and delivery channels. The events are described in the following sequence:

1. The Trading Partner A application begins the conversation by starting an instance of a private workflow which subsequently starts the collaborative workflow defined for the buyer role in the conversation. (This collaborative workflow is named `QueryPriceAvailability_Buyer`.)

2. The `QueryPriceAvailability_Buyer` collaborative workflow associates itself with the collaboration agreement `QueryPrice-A-A` (see Figure 2-9). In the `QueryPrice-A-A` collaboration agreement, the spoke delivery channel is the buyer and the routing proxy delivery channel is the supplier. Therefore the spoke delivery channel sends the message to the routing proxy delivery channel.

3. The routing proxy delivery channel is responsible for linking collaboration agreements. That is, a message arrives as part of one collaboration agreement and must be routed to another trading partner as part of another collaboration agreement. The collaboration agreements are linked because they use the same routing proxy delivery channel. (In this case they use `http://CompanyABC.com/xocp-hub-transport`.)

   When the routing proxy delivery channel receives this message, it is acting as the proxy supplier for the `QueryPrice-A-A` collaboration agreement. It then changes roles and becomes the proxy buyer. The Trading Partner A application

uses this common (routing proxy) delivery channel to find the collaboration agreements for the conversations in which it is acting as the buyer. In this case, the message is sent to the Supplier-Test's spoke delivery channel as part of the `QueryPrice-A-SpTest` collaboration agreement. Note that the routing proxy delivery channel is in the role of proxy buyer in the `QueryPrice-A-SpTest` collaboration agreement.

4. The Supplier-Test delivery channel uses the collaboration agreement ID in the message to identify the collaborative workflow for the supplier role, `QueryPriceAvailability_Supplier`. The message starts the `QueryPriceAvailability_Supplier` workflow.

5. The `QueryPriceAvailability_Supplier` collaborative workflow extracts the message and sends an event to a private workflow implemented by the Supplier-Test trading partner.

6. The private workflow transforms the request message to a response message and sends an event back to the `QueryPriceAvailability_Supplier` collaborative workflow.

7. The `QueryPriceAvailability_Supplier` workflow sends a reply message and terminates. Based on the roles defined in the `QueryPrice-A-SpTest` collaboration agreement (the Supplier-Test delivery channel is the supplier and the routing proxy delivery channel is the proxy buyer), the reply message is sent to the Trading Partner A routing proxy delivery channel.

8. Trading Party A again changes roles and becomes the proxy supplier. It uses the routing proxy delivery channel to find the collaboration agreements for the conversations in which it is acting as the buyer. In this case, the message is sent back to the Trading Partner A spoke delivery channel as part of the `QueryPrice-A-A` collaboration agreement.

9. The spoke delivery channel sends the message to the `QueryPriceAvailability_Buyer` collaborative workflow which, in turn, extracts the message and sends a *Done* event to the Trading Partner A private workflow. (The collaborative workflows are illustrated in Figure 2-7.)

Export and Import Repository Information

To participate in B2B collaborations, enterprises must be able to share information.

In the B2B example walkthrough described throughout this section, after Company ABC successfully tests its Query Price and Availability conversation (see "Send and Receive Business Messages" on page 2-18), it can prepare for production (that is, it can set up a Query Price and Availability conversation with Company XYZ).

The B2B Console provides an export facility that allows an enterprise to export data from its repository to an XML file. The XML file can then be imported into another WebLogic Integration repository, again using the B2B Console. You can select the scope of the exported file, that is, you can export all or a subset of the repository data. These export and import facilities are used by companies to share the information they need to set up B2B collaborations.

In our example, Company ABC completes the following tasks to prepare for business transactions with Company XYZ:

1. Import the Trading Partner Z data.

   Company XYZ must have generated the data for Trading Partner Z (as described in "Create Trading Partners and Delivery Channels" on page 2-13) and exported it to an XML file.

2. Use the B2B Console to change Trading Partner Z's *trading partner type* from *Local* to *Remote.*

3. Create a collaboration agreement between itself and Company XYZ.

   This is similar to the collaboration agreements described in "Create Collaboration Agreements" on page 2-16, except that the data for Company XYZ's Trading Partner Z is used, instead of the Supplier-Test trading partner and delivery channel data.

4. Export a copy of the newly created collaboration agreement using the B2B Console.

5. Export (publish) the `QueryPriceAvailability_Supplier` collaborative workflow template (see "Create Workflow Templates" on page 2-11) and document schemas from the WebLogic Integration Studio.

Company XYZ must subsequently complete the following tasks to prepare for business transactions with Company ABC:

1. Import the collaboration agreement prepared by Company ABC.
2. Use the B2B Console to change the following aspects of the data imported from Company ABC:
   - Trading Partner Z's *trading partner type* from *Local* to *Remote*
   - The Organization defined for Company ABC's role (`QueryPriceAvailability_Supplier`) in the Query Price and Availability conversation definition
3. Import the `QueryPriceAvailability_Supplier` collaborative workflow template and document schemas, which were published by Company XYZ from the WebLogic Integration Studio.
4. Creates a private workflow that works in conjunction with the `QueryPriceAvailability_Supplier` collaborative workflow. This private workflow implements the processes that occur locally to Trading Partner Z in its role as the supplier. These processes are not necessarily dictated by the conversation definition.

Start Business Collaboration

When the steps described in "Export and Import Repository Information" on page 2-21 are complete, each trading partner's repository has the information necessary to implement its role in the conversation definition.

The Trading Partner A application can begin the conversation by starting an instance of a private workflow, which subsequently starts the collaborative workflow defined for the buyer role in the conversation. (This collaborative workflow is named `QueryPriceAvailability_Buyer`.) Business messages are exchanged between Trading Partner A and Trading Partner Z applications, and the conversation progresses.

Index

D

E

F

H

I

J

L

M

N

P

Q

R

S

T

W

X

Z