



BEA WebLogic Integration™

Administering B2B Integration

Version 2.1
Document Date: October 2001

Copyright

Copyright © 2001 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic E-Business Platform, BEA WebLogic Enterprise, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Portal, BEA WebLogic Server and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Administering B2B Integration

Part Number	Date	Software Version
N/A	October 2001	2.1

Contents

About This Document

- What You Need to Know viii
- e-docs Web Site viii
- How to Print the Document ix
- Contact Us! ix
- Documentation Conventionsx

1. Configuration Requirements

- Configuration Overview 1-2
 - A Note About Trading Partner Encoding 1-5
- XOCP Applications 1-7
 - XOCP Hub and Spoke Delivery Channels 1-7
 - XOCP Peer-to-Peer Messaging 1-9
 - Trading Partners 1-9
 - Conversation Definitions 1-14
 - Collaboration Agreements 1-15
 - How It Works 1-16
 - XOCP Mediated Messaging 1-18
 - Trading Partners 1-19
 - Conversation Definitions 1-24
 - Collaboration Agreements 1-25
 - How It Works 1-28
- RosettaNet Applications 1-30
 - Trading Partners 1-31
 - Conversation Definitions 1-34
 - Collaboration Agreements 1-35
 - How It Works 1-35
- cXML Applications 1-37

Browser Clients	1-40
Hosting a Browser Client	1-43
File-Sharing Clients	1-46
Hosting a File-Sharing Client.....	1-47
2. Basic Configuration Tasks	
Overview of WebLogic Integration B2B Console	2-2
Getting Help	2-7
Configuring the B2B Engine	2-8
Configuring Trading Partners	2-9
Configuring Conversation Definitions	2-13
Configuring Collaboration Agreements	2-15
3. Advanced Configuration Tasks	
Overview of Advanced Features	3-2
XPath Expressions in Routing and Filtering	3-3
XPath Router Expression Processing	3-6
XPath Filter Expression Processing	3-7
Configuring Router and Filter Expressions	3-7
Additional Information	3-9
Custom Logic Plug-Ins	3-9
Configuring Logic Plug-Ins.....	3-11
Adding a Custom Logic Plug-Into a Business Protocol Router or Filter Chain	3-11
Additional Information	3-12
Trading Partner Extended Properties	3-13
Configuring Trading Partner Extended Properties	3-14
Additional Information	3-15
4. Importing and Exporting B2B Integration Components	
B2B Integration Components	4-2
Export and Import Overview	4-3
Exporting from the B2B Console	4-4
Importing to the B2B Console	4-8
Exporting a Workflow Package.....	4-9
Importing a Workflow Package.....	4-12

5. Monitoring B2B Integration

Overview of Monitoring.....	5-2
Note About Conversation Monitoring.....	5-3
B2B Console Monitoring Pages.....	5-4
Monitoring the B2B Engine.....	5-6
Monitoring Trading Partner Sessions.....	5-9
Monitoring Delivery Channels.....	5-11
Monitoring Conversations.....	5-12
Monitoring Collaboration Agreements.....	5-13
Monitoring Messages.....	5-16

6. Working with the Repository

Understanding the Repository.....	6-1
B2B Configuration Elements.....	6-2
Managing the B2B Configuration Information in the Repository.....	6-7

7. Working with the Bulk Loader

Understanding the Terminology.....	7-2
Importing Data into the Repository.....	7-3
How the Bulk Loader Imports Data.....	7-3
Procedure for Importing Data into the Repository.....	7-6
Exporting Data from the Repository.....	7-7
How the Bulk Loader Exports Data.....	7-8
Full and Partial Repository Exports.....	7-9
Short and Long Repository Exports.....	7-10
Procedure for Exporting Repository Data.....	7-11
Deleting Data from the Repository.....	7-12
How the Bulk Loader Deletes Data.....	7-12
Procedure for Deleting Repository Data.....	7-13
Working with the Bulk Loader Configuration File.....	7-14
Bulk Loader Configuration File for Importing Data.....	7-18
Bulk Loader Configuration File for Exporting Data.....	7-20
Working with the Repository Data File.....	7-21
Checking Data.....	7-23
Creating an Error Log.....	7-23

Validating XML Files.....	7-24
Checking Data Integrity	7-24
Checking Data Integrity While Importing or Deleting	7-25
Checking Data Integrity During an Export	7-26
Forcing the Bulk Loader.....	7-26

8. Configuring Persistence and Recovery

Understanding Persistence and Recovery.....	8-2
Understanding Persistent Mode.....	8-3
Understanding Nonpersistent Mode	8-4
Understanding Recovery	8-5
Configuring B2B Engine Startup	8-6
Configuring Persistence and Recovery.....	8-9

A. Update Considerations

About This Document

This document describes how to configure and manage BEA WebLogic Integration business-to-business (B2B) integration solutions.

Specifically, it discusses the following topics:

- Chapter 1, “Configuration Requirements,” presents sample configurations that summarize the requirements for typical B2B integration solutions.
- Chapter 2, “Basic Configuration Tasks,” provides an overview of the tasks and procedures required to configure the B2B engine for trading exchange, supply chain management, and collaborative commerce applications.
- Chapter 3, “Advanced Configuration Tasks,” serves as an introduction to the more advanced features related to B2B integration and provides a roadmap to where more detailed information can be found.
- Chapter 4, “Importing and Exporting B2B Integration Components,” describes how you can export and import components to facilitate implementation of B2B integration solutions.
- Chapter 5, “Monitoring B2B Integration,” provides an overview of how you can use the WebLogic Integration B2B Console to monitor and control trading partner sessions, delivery channels, conversations, and collaboration agreements.
- Chapter 6, “Working with the Repository,” describes how B2B integration configuration elements are stored in the WebLogic Integration repository.
- Chapter 7, “Working with the Bulk Loader,” provides the information you need to use the Bulk Loader to load data into the repository.
- Chapter 8, “Configuring Persistence and Recovery,” describes how persistence and recovery are supported by the B2B engine.
- Chapter A, “Update Considerations,” summarizes the constraints and considerations related to modifying B2B integration configuration elements.

What You Need to Know

This document is intended mainly for developers and system administrators responsible for the creation, setup, and administration of trading exchange, supply chain management, and collaborative commerce applications deployed on BEA WebLogic Integration.

Before reading this document, we recommend that you read the following documents:

- *Introducing BEA WebLogic Integration*
- *Introducing B2B Integration*
- *Learning to Use BEA WebLogic Integration*
- *Starting, Stopping, and Customizing BEA WebLogic Integration*

Before you begin designing your own B2B integration solutions, we recommend that you read the following documents:

- *Designing BEA WebLogic Integration Solutions*
- *Deploying BEA WebLogic Integration Solutions*

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” 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 from the BEA WebLogic Integration documentation Home page, which is available on the documentation CD and on the e-docs Web site at <http://e-docs.bea.com>. 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 BEA WebLogic Integration documentation Home page, click the PDF Files button, and select the document you want to print.

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

Contact Us!

Your feedback on the BEA 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 the BEA WebLogic Integration 2.1 release.

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

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
<i>italics</i>	Indicates emphasis or book titles.
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. <i>Examples:</i> <pre>#include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</pre>

Convention	Item
monospace boldface text	Identifies significant words in code. <i>Example:</i> void commit ()
<i>monospace</i> <i>italic</i> <i>text</i>	Identifies variables in code. <i>Example:</i> String <i>expr</i>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> 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. <i>Example:</i> buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: <ul style="list-style-type: none"> ■ 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. <i>Example:</i> buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.



1 Configuration Requirements

This section presents selected WebLogic Integration B2B integration configurations and provides examples that summarize the configuration requirements for typical participants. It includes the following topics:

- Configuration Overview
- XOCP Applications
- RosettaNet Applications
- cXML Applications
- Browser Clients
- File-Sharing Clients

The information in this section applies to configurations for collaborative workflows used to manage conversations between trading partners. If you are using the Messaging API or the cXML API to write Java messaging applications, see [Programming Messaging Applications for B2B Integration](#). If you are using logic plug-ins to add functionality to a collaboration, see [Programming Logic Plug-Ins for B2B Integration](#).

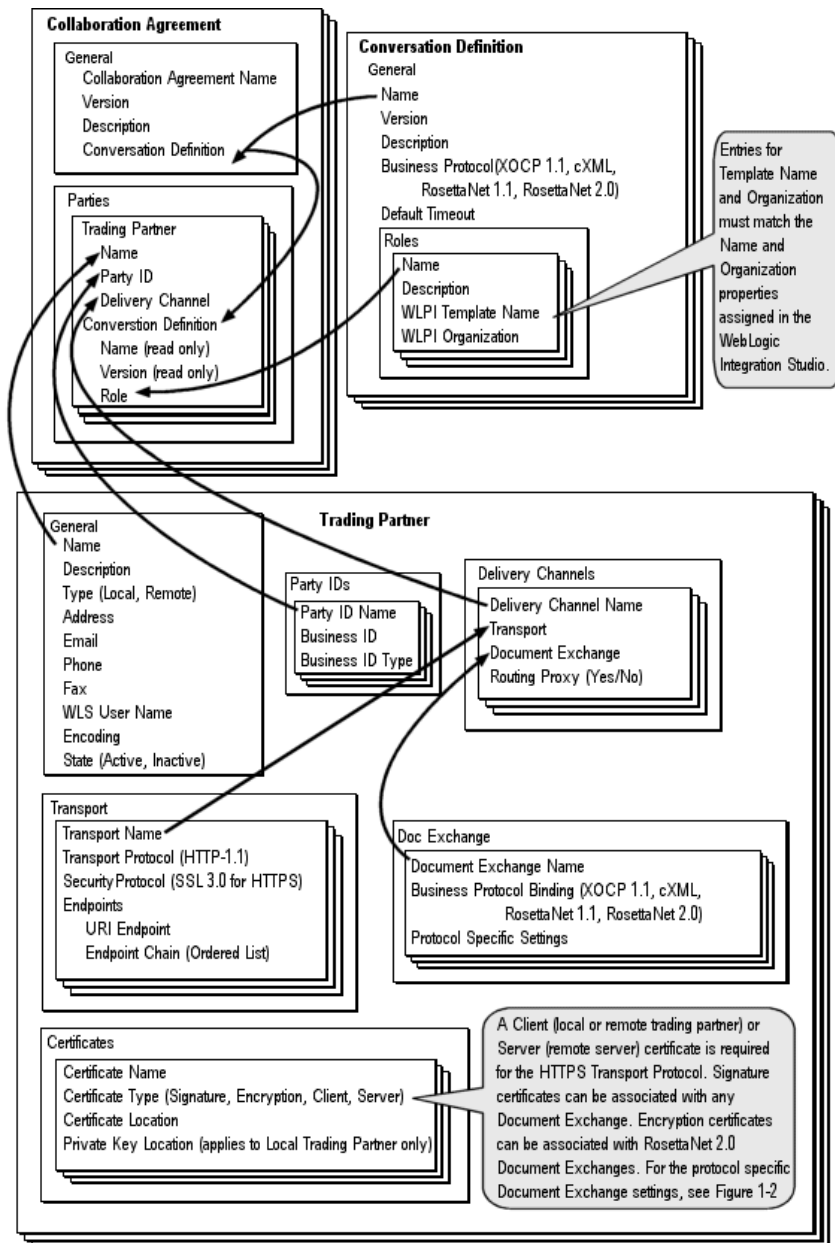
Configuration Overview

The following sections describe how to set up some of the basic B2B integration applications supported by WebLogic Integration. These basic configurations illustrate the concepts required for more complex scenarios. In addition to setting the B2B engine properties (for example, B2B engine name, description, and preferences), properties for the following entities must be configured:

- *Trading partners*
Basic identifying information, party identifiers, and delivery channel information must be defined for each participant.
- *Conversation definitions*
Each conversation definition must include a name, a version, an assigned business protocol, and two or more roles, each associated with a workflow template.
- *Collaboration agreements*
A collaboration agreement specifies a conversation definition and the parties to the agreement. For each party, a trading partner party identifier, delivery channel, and role must be defined.

The following figure provides an overview of the entities that must be configured. It shows how information is organized in the WebLogic Integration B2B Console. For a summary of how information is stored in the repository, see Chapter 6, “Working with the Repository.” The detailed information required to set specific properties is provided in the B2B Console online help (see “Getting Help” on page 2-7).

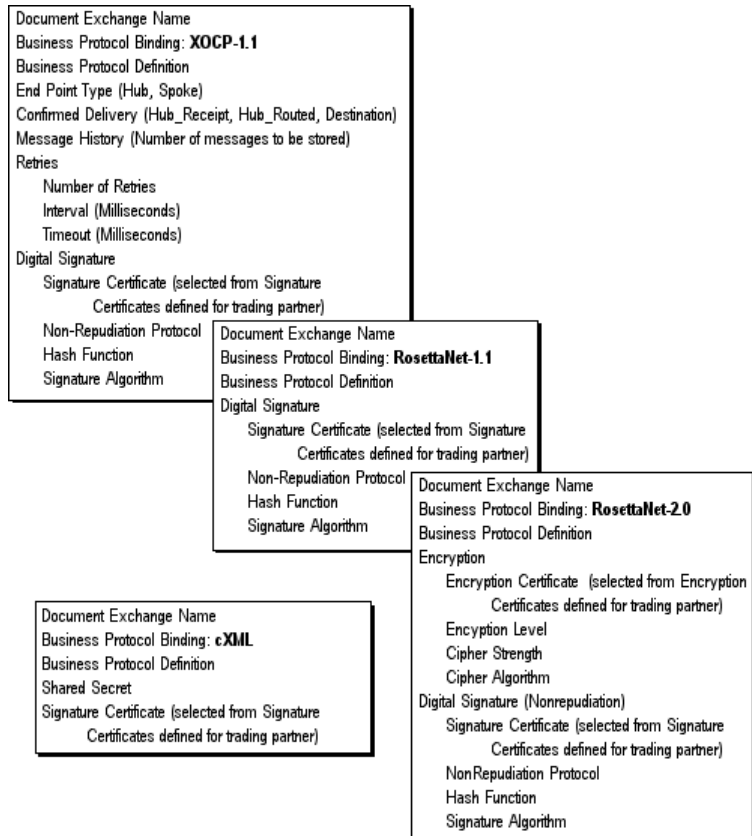
Figure 1-1 B2B Integration Configuration Overview



1 Configuration Requirements

As shown in the preceding figure, each delivery channel is associated with a document exchange, which defines its business protocol binding. The parameters defined in the document exchange vary, depending on the selected protocol. The following figure summarizes the four protocols supported by WebLogic Integration.

Figure 1-2 Business Protocol Bindings Defined in the Document Exchange



The following sections describe the configuration requirements for participants in several sample applications. Although configuration is typically performed through the B2B Console, these examples do not demonstrate how configuration is done. Keep in mind that the trading partners, conversation definitions, collaboration agreements, and workflows created on one system can be exported and then imported to another system, as described in Chapter 4, “Importing and Exporting B2B Integration Components.”

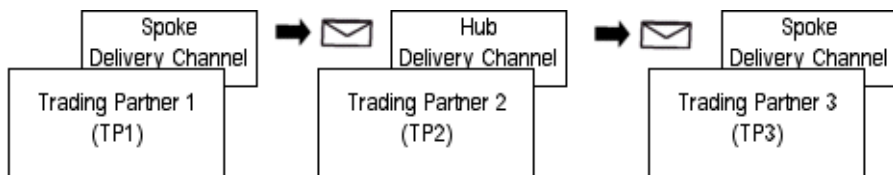
A Note About Trading Partner Encoding

Like all XML documents, the messages exchanged between trading partners can potentially be encoded using any valid character set. By default, messages are encoded in UTF-8. To support alternate encodings, the configuration for a trading partner includes the property, *encoding*, which determines the how the B2B engine encodes messages sent to that trading partner. (See the trading partner general properties shown in Figure 1-1.)

The specification of encoding only has significance in the XOCP protocol. The B2B engine uses UTF-8 to encode all cXML and RosettaNet messages, regardless of how the encoding property is set.

As you will learn in the following section, “XOCP Applications,” messages in the XOCP protocol are always routed though a trading partner delivery channel that is configured as a hub, as shown in the following figure.

Figure 1-3 Message Delivery



Suppose the following encoding settings applied to the trading partners shown in the preceding figure:

- TP1, encoding is not set (the default of UTF-8 is used)
- TP2, encoding=UTF-16
- TP3, encoding=Shift_JIS

Based on the these settings, encoding of a message from TP1 destined for TP3 would be handled as follows:

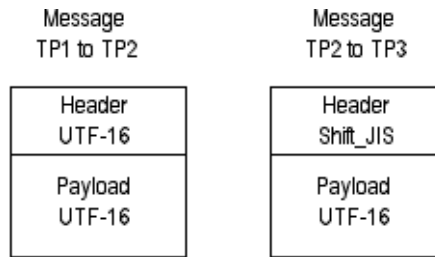
1. TP1 sends message to TP2. Based on the encoding set for TP2, the message is encoded in UTF-16.
2. TP2 receives the message. Because the delivery channel on TP2 is a hub delivery channel, message processing only involves the message header. The payload remains unchanged.

1 Configuration Requirements

3. TP2 sends the message to TP3. The header is encoded in conformity with the encoding setting for TP3. The payload remains unchanged as noted.

The message encoding for this scenario is summarized in the following figure.

Figure 1-4 Message Encoding



In order to achieve the desired results when using the encoding property, it is important to keep in mind how the message will be handled when you configure the trading partners and their associated delivery channels.

If the intention is to have the payload information encoded in Shift_JIS for TP3, then the encoding for TP2 should be set to Shift_JIS.

The following section provides additional information about how trading partner definitions, conversation definitions, and collaboration agreements are configured to route messages using the XOCP protocol. If required, you can configure dummy trading partners and collaboration agreements as needed to ensure trading partners receive messages encoded appropriately.

For a list of valid encoding names and aliases, visit the following URL:

<http://www.iana.org/assignments/character-sets>

XOCP Applications

The following sections provide background information about the configuration of XOCP delivery channels and examples that show how to configure the B2B engine to use XOCP in peer-to-peer and mediated messaging applications:

- XOCP Hub and Spoke Delivery Channels
- XOCP Peer-to-Peer Messaging
- XOCP Mediated Messaging

XOCP Hub and Spoke Delivery Channels

A delivery channel in the XOCP protocol must be configured as one of the following:

- A hub (routing proxy) delivery channel
- A spoke delivery channel

A hub delivery channel is unique in that it acts as a proxy for either role in a conversation definition. The role assumed by the hub delivery channel depends on the configuration of the collaboration agreements for the conversation definition. For example, if the hub delivery channel for a trading partner is assigned to the role of supplier in one collaboration agreement, and to the role of buyer in another collaboration agreement for the same conversation definition, messages are handled as follows:

- The hub delivery channel receives a message sent to the supplier role. All collaboration agreements for the conversation definition in which this hub delivery channel is assigned the buyer role are identified. The message is forwarded to the trading partner delivery channel assigned to the supplier role in each of these collaboration agreements.
- The hub delivery channel receives a message sent to the buyer role. All collaboration agreements for the conversation definition in which this hub delivery channel is assigned the supplier role are identified. The message is forwarded to the trading partner delivery channel assigned to the buyer role in each of these collaboration agreements.

1 Configuration Requirements

In other words, if multiple trading partners associated with the same role in a conversation definition enter into a collaboration agreement with a hub delivery channel, that delivery channel can be used to broadcast messages to those trading partners. If it is necessary to limit the broadcast to a subset of trading partners, XPath filter expressions can be used, as described in “XPath Expressions in Routing and Filtering” on page 3-3.

Although XOCP applications always require a hub-and-spoke configuration for their delivery channels and collaboration agreements, trading partners using the XOCP protocol can participate in either a peer-to-peer or mediated exchange of messages:

- *Peer-to-peer message exchange*

Each participating trading partner has a role in the collaboration; messages are exchanged between trading partner peers. To accomplish this, one of the trading partner peers must configure two delivery channels: one as a hub delivery channel; the other, as a spoke delivery channel. Please note that we are simulating the peer-to-peer model here; the XOCP point-to-point protocol is not available in the current version of the product.

- *Mediated message exchange*

A trading partner acting as a mediator does not participate directly in any role; instead, it routes messages to the trading partner spokes. The mediator configures a hub delivery channel and each trading partner spoke defines a spoke delivery channel. On the mediator, a collaboration agreement is defined between the trading partner hub delivery channel and each trading partner spoke delivery channel. The collaboration agreement assigns the hub and spoke delivery channels to roles as required for proper routing of messages.

Detailed examples of these two basic XOCP message exchange methods are presented in the following sections. Keep in mind that while these examples themselves show simple scenarios, the principles they illustrate can be used to create more complex hybrid deployments.

XOCP Peer-to-Peer Messaging

Suppose two trading partners, ABC International, a computer manufacturer, and XYZ Systems, a chip supplier, plan to use WebLogic Integration to participate in Query Price and Availability (QPA) transactions. ABC International is the buyer and the initiator of each transaction, and the XOCP protocol is used to exchange messages. Both trading partners have WebLogic Integration installed.

The following sections provide examples of how to:

- Configure the trading partners and their associated delivery channels
- Configure a conversation definition to implement the required roles
- Associate the required trading partner delivery channels with the appropriate roles in the collaboration agreements

It is assumed that the required private and collaborative (public) workflows have been created. The collaborative workflows in this example are named `QPA_Public_Supplier` and `QPA_Public_Buyer`. For information about using the Studio (with the extended functionality provided by the B2B integration plug-in) to create collaborative workflows, see [Creating Workflows for B2B Integration](#).

Caution: This example demonstrates a *single* trading partner configured with *two* delivery channels: one hub, one spoke. Please note that there is currently a limitation with this configuration. Do not set up one trading partner with two delivery channels. Instead set up two trading partners, each with its own delivery channel. Configure a hub delivery channel for one trading partner, and a spoke delivery channel for the other.

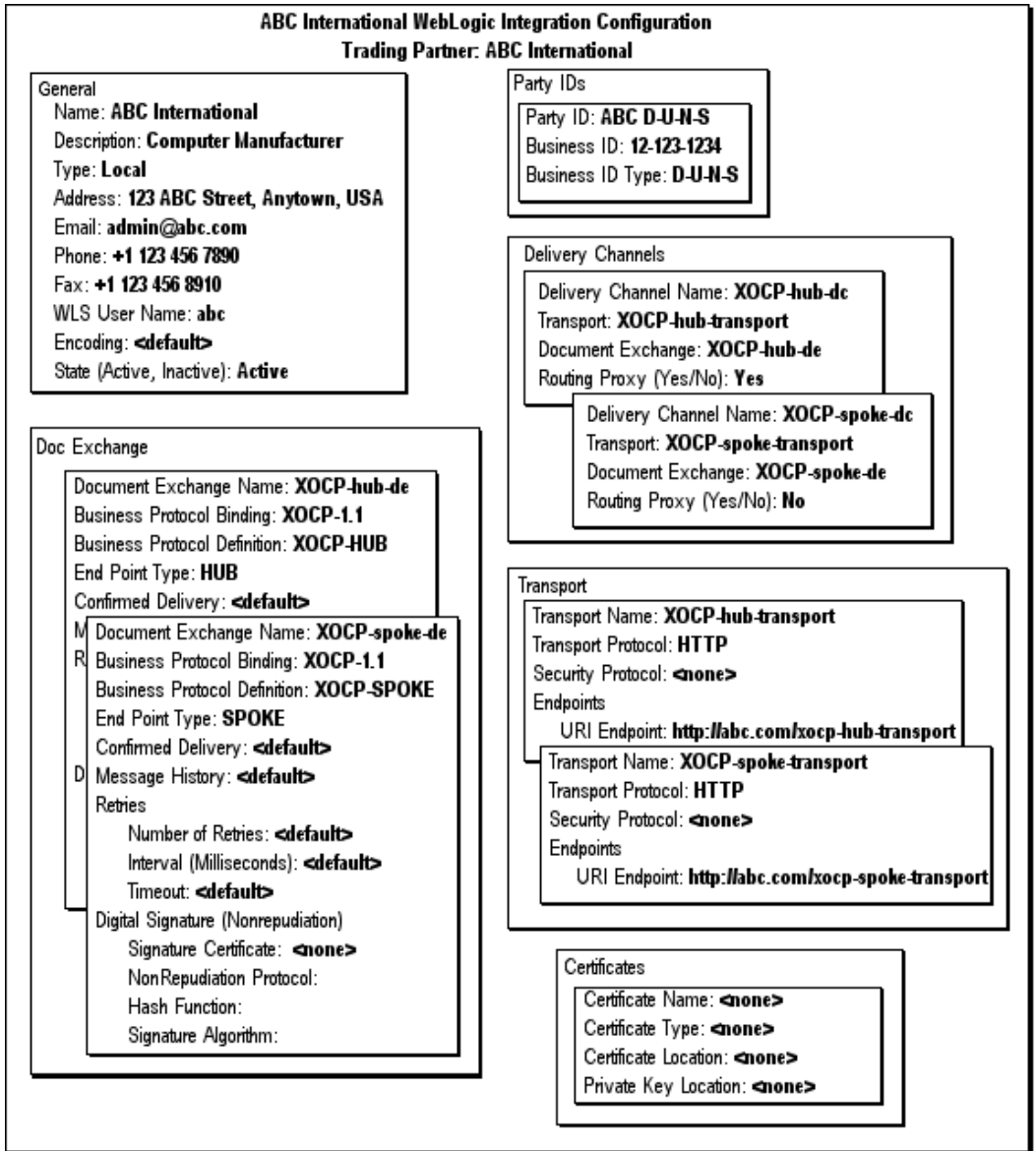
Trading Partners

Trading partner definitions for both ABC International and XYZ Systems must be provided.

Note: For simplicity, SSL or signature certificates are not used in the examples presented in this section. For information about security configuration, see [Implementing Security with B2B Integration](#).

The information that must be defined for the ABC International trading partner in the ABC WebLogic Integration configuration is summarized in the following figure.

Figure 1-5 ABC International Trading Partner Definition in the ABC Configuration



Note that the trading partner type (listed in the General box) is `Local` and the configuration includes hub and spoke delivery channels (`XOCP-hub-dc` and `XOCP-spoke-dc`), document exchanges (`XOCP-hub-de` and `XOCP-spoke-de`), and transports (`XOCP-hub-transport` and `XOCP-spoke-transport`). Each transport is associated with a URI that serves as the delivery channel endpoint (`http://abc.com/xocp-hub-transport` and `http://abc.com/xocp-spoke-transport`).

Note: As noted in the introduction to this example (see “XOCP Peer-to-Peer Messaging” on page 1-9), WebLogic Integration currently has a limitation that requires you to configure two trading partners. For example, configure one trading partner named `ABC International-Spoke` and the other named `ABC International`. The `ABC International` definition would include the `XOCP-hub-dc`, `XOCP-hub-de`, and `XOCP-hub-transport` definitions, while the `ABC International-Spoke` definition would include the `XOCP-spoke-dc`, `XOCP-spoke-de`, and `XOCP-spoke-transport` definitions.

The information that must be defined for the `ABC International` trading partner in the XYZ WebLogic Integration configuration is summarized in the following figure.

1 Configuration Requirements

Figure 1-6 ABC International Trading Partner Definition in the XYZ Configuration

XYZ Systems WebLogic Integration Configuration
Trading Partner: ABC International

General Name: ABC International Description: Computer Manufacturer Type: Remote Address: 123 ABC Street, Anytown, USA Email: admin@abc.com Phone: +1 123 456 7890 Fax: +1 123 456 8910 WLS User Name: abc Encoding: <default> State (Active, Inactive): Active	Party IDs Party ID: ABC D-U-I-S Business ID: 12-123-1234 Business ID Type: D-U-I-S
Doc Exchange Document Exchange Name: XOCP-hub-de Business Protocol Binding: XOCP-1.1 Business Protocol Definition: XOCP-HUB End Point Type: HUB Confirmed Delivery: <default> Message History: <default> Retries Number of Retries: <default> Interval (Milliseconds): <default> Timeout: <default> Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol: Hash Function: Signature Algorithm:	Delivery Channels Delivery Channel Name: XOCP-hub-dc Transport: XOCP-hub-transport Document Exchange: XOCP-hub-de Routing Proxy (Yes/No): Yes
	Transport Transport Name: XOCP-hub-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://abc.com/xocp-hub-transport
	Certificates Certificate Name: <none> Certificate Type: <none> Certificate Location: <none>

In this case the trading partner type (listed in the General box) is `Remote` and the configuration includes only a hub delivery channel (`XOCP-hub-dc`), document exchange (`XOCP-hub-de`), and transport (`XOCP-hub-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://abc.com/xocp-hub-transport`).

The information that must be defined for the XYZ Systems trading partner in the ABC WebLogic Integration configuration is summarized in the following figure.

Figure 1-7 XYZ Systems Trading Partner Definition in the ABC International Configuration

ABC International WebLogic Integration Configuration
Trading Partner: XYZ Systems

<p>General Name: XYZ Systems Description: Chip Supplier Type: Remote Address: 456 XYZ Street, Anytown, USA Email: admin@xyz.com Phone: +1 456 789 1020 Fax: +1 456 789 1021 WLS User Name: xyz Encoding: <default> State (Active, Inactive): Active</p>	<p>Party IDs Party ID: XYZ D-U-N-S Business ID: 34-567-8910 Business ID Type: D-U-N-S</p>
<p>Doc Exchange Document Exchange Name: XOCP-spoke-de Business Protocol Binding: XOCP-1.1 Business Protocol Definition: XOCP-SPOKE End Point Type: SPOKE Confirmed Delivery: <default> Message History: <default> Retries Number of Retries: <default> Interval (Milliseconds): <default> Timeout: <default> Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol: Hash Function: Signature Algorithm:</p>	<p>Delivery Channels Delivery Channel Name: XOCP-spoke-dc Transport: XOCP-spoke-transport Document Exchange: XOCP-spoke-de Routing Proxy (Yes/No): No</p>
	<p>Transport Transport Name: XOCP-spoke-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://xyz.com/xocp-spoke-transport</p>
	<p>Certificates Certificate Name: <none> Certificate Type: <none> Certificate Location: <none></p>

1 Configuration Requirements

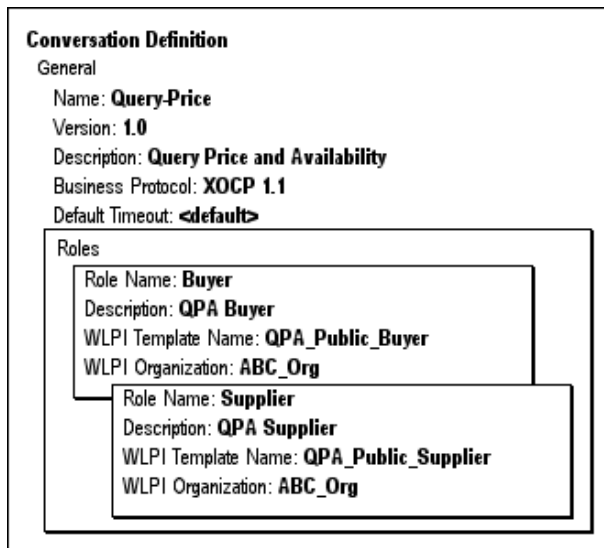
In this case the trading partner type (listed in the General box) is `Remote` and the configuration includes only a spoke delivery channel (`XOCP-spoke-dc`), document exchange (`XOCP-spoke-de`), and transport (`XOCP-spoke-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://xyz.com/xocp-spoke-transport`).

The information that must be defined for the XYZ Systems trading partner in the XYZ WebLogic Integration configuration is exactly the same as the information shown in the previous figure, with one exception: the trading partner type (listed in the General box) must be set to `Local` in the XYZ WebLogic Integration configuration.

Conversation Definitions

The required settings for the Query Price and Availability (QPA) conversation definition in the ABC WebLogic Integration configuration are shown in the following figure.

Figure 1-8 Conversation Definition for QPA



The conversation definition in the XYZ WebLogic Integration configuration is the same, with one exception: the value of the BPM organization (listed as WLPI organization) must be changed to reflect an organization defined at XYZ systems (for example, `XYZ_Org`).

Collaboration Agreements

The collaboration agreement shown in the following figure is required only in the ABC WebLogic Integration configuration. The agreement makes it possible for the hub delivery channel to act as a proxy for the buyer role, as described in “XOCP Hub and Spoke Delivery Channels” on page 1-7.

Figure 1-9 Collaboration Agreement ABC-ABC

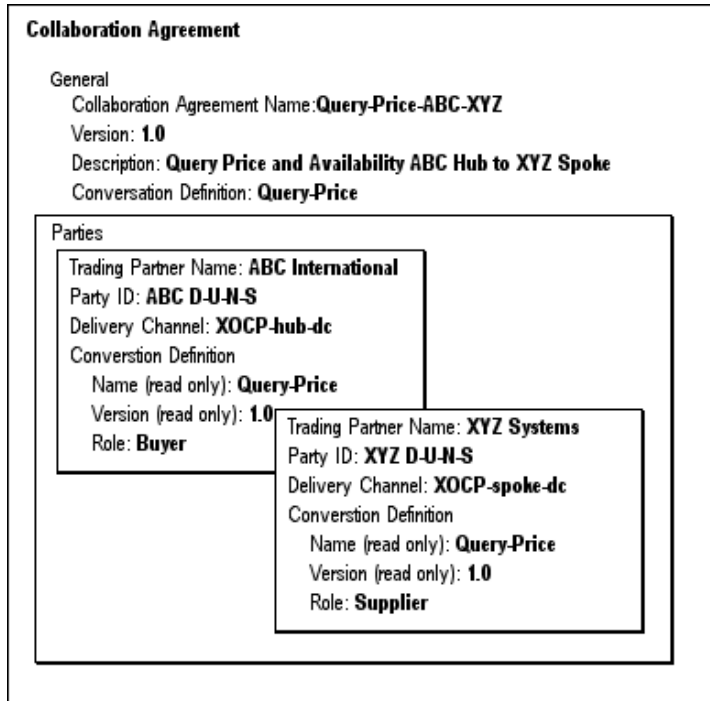
Collaboration Agreement	
General	
Collaboration Agreement Name: Query-Price-ABC-ABC	
Version: 1.0	
Description: Query Price and Availability ABC Hub to ABC Spoke	
Conversation Definition: Query-Price	
Parties	
Trading Partner Name: ABC International Party ID: ABC D-U-N-S Delivery Channel: XOCP-hub-dc Conversation Definition Name (read only): Query-Price Version (read only): 1.0 Role: Supplier	Trading Partner Name: ABC International Party ID: ABC D-U-N-S Delivery Channel: XOCP-spoke-dc Conversation Definition Name (read only): Query-Price Version (read only): 1.0 Role: Buyer

Note: If you have created two trading partners as described in the note following Figure 1-5, the trading partner name for the buyer role in the previous collaboration agreement is changed to ABC International-Spoke.

1 Configuration Requirements

The collaboration agreement shown in the following figure is required in both the ABC and XYZ WebLogic Integration configurations.

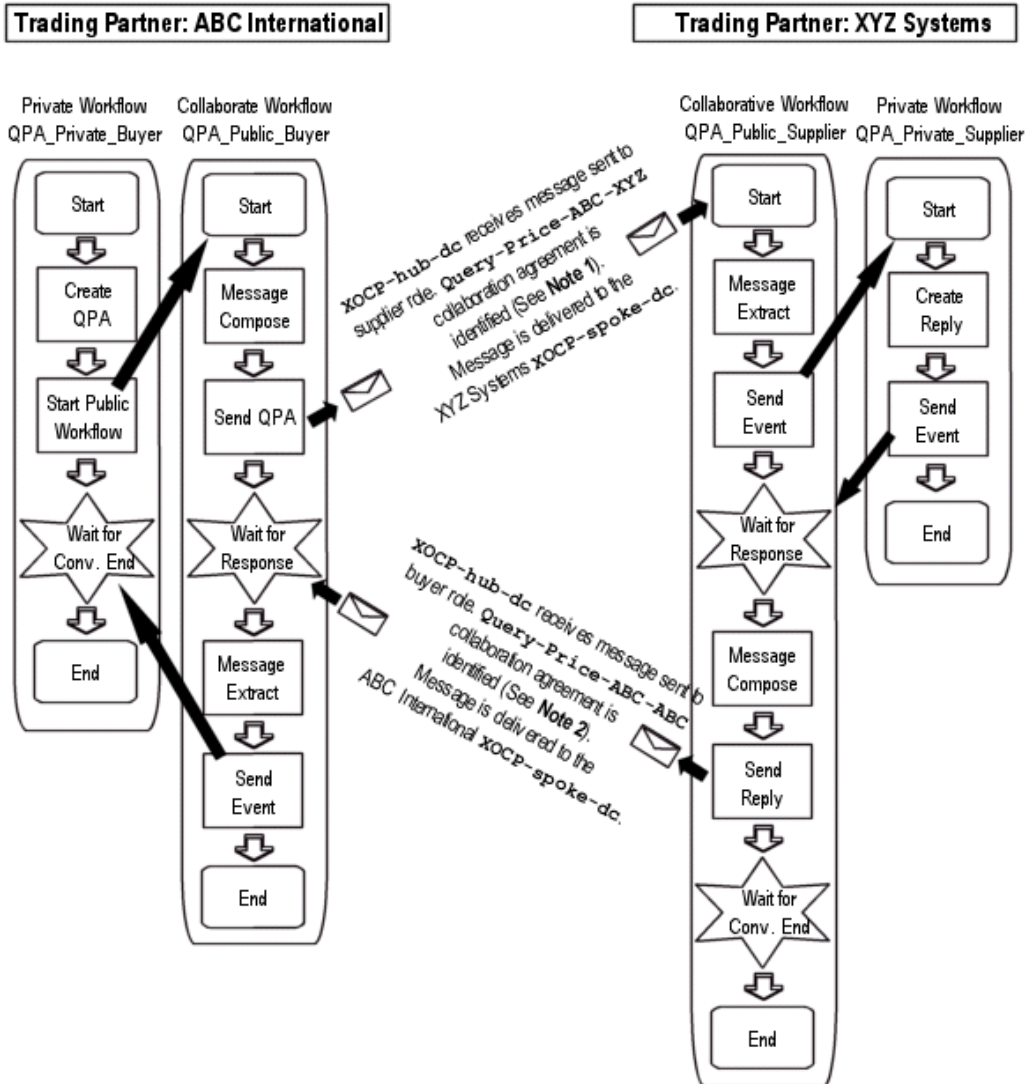
Figure 1-10 Collaboration Agreement ABC-XYZ



How It Works

The following figure shows how WebLogic Integration supports the exchange of Query Price and Availability (QPA) messages by ABC International and XYZ Systems.

Figure 1-11 QPA Collaboration Overview: XOCP Peer-to-Peer



1 Configuration Requirements

Note 1

The ABC International hub delivery channel (XOCP-hub-dc) receives the message sent to the supplier role. As described in “XOCP Hub and Spoke Delivery Channels” on page 1-7, all collaboration agreements in which XOCP-hub-dc is assigned the buyer role are identified. In the case shown here, one collaboration agreement, Query-Price ABC-XYZ, meets the criteria. Based on that agreement, the message is delivered to the XYZ Systems delivery channel assigned to the role of supplier (XYZ Systems XOCP-spoke-dc at <http://xyz.com/xocp-spoke-transport>).

Note 2

The ABC International hub delivery channel (XOCP-hub-dc) receives the message sent to the buyer role. All collaboration agreements in which XOCP-hub-dc is assigned the role of supplier are identified. In the case shown here, one collaboration agreement, Query-Price ABC-ABC, meets the criteria. Based on that agreement, the message is delivered to the XYZ Systems delivery channel assigned to the role of buyer (ABC International XOCP-spoke-dc at <http://abc.com/xocp-spoke-transport>).

XOCP Mediated Messaging

In this example, ABC International, a computer manufacturer, plans to contract with IntCo, a company that acts as an intermediary for order management transactions. Two chip suppliers, TUV Corporation and XYZ Systems, are among those contracted with IntCo.

IntCo acts as the intermediary in the Query Price and Availability (QPA) transactions between ABC International and the two chip suppliers. IntCo does not directly participate in any role in the conversation, but acts as a mediator in the transactions between ABC International and the suppliers.

ABC International is the buyer and the initiator of each transaction and the XOCP protocol is used to exchange messages. All participants have WebLogic Integration installed.

The following sections provide examples of how to:

- Configure the trading partners and their associated delivery channels
- Configure a conversation definition to implement the required roles
- Associate the required trading partner delivery channels with the roles defined in the collaboration agreements

It is assumed that the required private and collaborative (public) workflows have been created. The collaborative workflows in this example are named `QPA_Public_Supplier` and `QPA_Public_Buyer`. For information about using the Studio (with the extended functionality provided by the B2B integration plug-in) to create collaborative workflows, see [Creating Workflows for B2B Integration](#).

Trading Partners

Trading partners must be configured as follows:

- ABC International and IntCo must be defined in the ABC WebLogic Integration configuration.
- TUV Corporation and IntCo must be defined in the TUV WebLogic Integration configuration.
- XYZ Systems and IntCo must be defined in the XYZ Systems WebLogic Integration configuration.
- ABC International, TUV Corporation, XYZ Systems, and IntCo must be defined in the IntCo WebLogic Integration configuration.

Note: For simplicity, SSL or signature certificates are not used in the examples in this section. For information about security configuration, see [Implementing Security with B2B Integration](#).

The IntCo trading partner definition required in the ABC, TUV, and XYZ WebLogic Integration configurations is summarized in the following figure.

Figure 1-12 IntCo Trading Partner Definition in the ABC, TUV, and XYZ Configurations

ABC, TUV, and XYZ WebLogic Integration Configurations
Trading Partner: IntCo

General Name: IntCo Description: Order Management Type: Remote Address: 1 Intco Plaza, Anytown, USA Email: admin@intco.com Phone: +1 678 910 1112 Fax: +1 678 910 1113 WLS User Name: intco Encoding: <default> State (Active, Inactive): Active	Party IDs Party ID: Intoc D-U-N-S Business ID: 10-234-5678 Business ID Type: D-U-N-S
Doc Exchange Document Exchange Name: XOCP-hub-de Business Protocol Binding: XOCP-1.1 Business Protocol Definition: XOCP-HUB End Point Type: HUB Confirmed Delivery: <default> Message History: <default> Retries Number of Retries: <default> Interval (Milliseconds): <default> Timeout: <default> Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol: Hash Function: Signature Algorithm:	Delivery Channels Delivery Channel Name: XOCP-hub-dc Transport: XOCP-hub-transport Document Exchange: XOCP-hub-de Routing Proxy (Yes/No): Yes
	Transport Transport Name: XOCP-hub-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://intco.com/xocp-hub-transport
	Certificates Certificate Name: <none> Certificate Type: <none> Certificate Location: <none>

Note that the trading partner type (listed in the General box) is `Remote` and the configuration includes a hub delivery channel (`XOCP-hub-dc`), document exchange (`XOCP-hub-de`), and transport (`XOCP-hub-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://intco.com/xocp-hub-transport`).

The IntCo trading partner definition in the IntCo WebLogic Integration configuration must match the definition shown in the previous figure, with one exception: the trading partner type (listed in the General box) must be set to `Local` in the IntCo WebLogic Integration configuration.

Each of the other trading partners must provide a trading partner definition for itself as part of its own WebLogic Integration configuration.

The TUV Corporation trading partner definition required in the TUV Corporation configuration is summarized in the following figure.

Figure 1-13 TUV Corporation Trading Partner Definition in the TUV Configuration

TUV Corporation WebLogic Collaborate Configuration
Trading Partner: **TUV Corporation**

<p>General Name: TUV Corporation Description: Chip Supplier Type: Local Address: 789 TUV Street, Anytown, USA Email: admin@tuv.com Phone: +1 567 890 1234 Fax: +1 567 890 2345 WLS User Name: tuv State (Active, Inactive): Active</p>	<p>Party IDs Party ID: TUV D-U-N-S Business ID: 45-678-9012 Business ID Type: D-U-N-S</p>
<p>Doc Exchange Document Exchange Name: XOCP-spoke-de Business Protocol Binding: XOCP-1.1 Business Protocol Definition: XOCP-SPOKE End Point Type: SPOKE Confirmed Delivery: <default> Message History: <default> Retries Number of Retries: <default> Interval (Milliseconds): <default> Timeout: <default> Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol: Hash Function: Signature Algorithm:</p>	<p>Delivery Channels Delivery Channel Name: XOCP-spoke-dc Transport: XOCP-spoke-transport Document Exchange: XOCP-spoke-de Routing Proxy (Yes/No): No</p>
	<p>Transport Transport Name: XOCP-spoke-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://tuv.com/xocp-spoke-transport</p>
	<p>Certificates Certificate Name: <none> Certificate Type: <none> Certificate Location: <none></p>

1 Configuration Requirements

Note that the trading partner type is `Local` and the configuration includes a spoke delivery channel (`XOCP-spoke-dc`), document exchange (`XOCP-spoke-de`), and transport (`XOCP-spoke-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://tuv.com/xocp-spoke-transport`).

The XYZ Systems trading partner definition required in the XYZ configuration is summarized in the following figure.

Figure 1-14 XYZ Systems Trading Partner Definition in the XYZ Configuration

XYZ Systems WebLogic Integration Configuration
Trading Partner: XYZ Systems

General Name: XYZ Systems Description: Chip Supplier Type: Local Address: 456 XYZ Street, Anytown, USA Email: admin@xyz.com Phone: +1 456 789 1020 Fax: +1 456 789 1021 WLS User Name: xyz Encoding: <default> State (Active, Inactive): Active	Party IDs Party ID: XYZ D-U-N-S Business ID: 34-567-8910 Business ID Type: D-U-N-S
Doc Exchange Document Exchange Name: XOCP-spoke-de Business Protocol Binding: XOCP-1.1 Business Protocol Definition: XOCP-SPOKE End Point Type: SPOKE Confirmed Delivery: <default> Message History: <default> Retries Number of Retries: <default> Interval (Milliseconds): <default> Timeout: <default> Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol: Hash Function: Signature Algorithm:	Delivery Channels Delivery Channel Name: XOCP-spoke-dc Transport: XOCP-spoke-transport Document Exchange: XOCP-spoke-de Routing Proxy (Yes/No): No
	Transport Transport Name: XOCP-spoke-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://xyz.com/xocp-spoke-transport
	Certificates Certificate Name: <none> Certificate Type: <none> Certificate Location: <none>

Note that the trading partner type is `Local` and the configuration includes a spoke delivery channel (`XOCP-spoke-dc`), document exchange (`XOCP-spoke-de`), and transport (`XOCP-spoke-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://xyz.com/xocp-spoke-transport`).

The ABC International trading partner definition required in the ABC configuration is summarized in the following figure.

Figure 1-15 ABC International Trading Partner Definition in the ABC Configuration

ABC International WebLogic Integration Configuration	
Trading Partner: ABC International	
<p>General</p> <p>Name: ABC International Description: Computer Manufacturer Type: Local Address: 123 ABC Street, Anytown, USA Email: admin@abc.com Phone: +1 123 456 7890 Fax: +1 123 456 8910 WLS User Name: abc Encoding: <default> State (Active, Inactive): Active</p>	<p>Party IDs</p> <p>Party ID: ABC D-U-N-S Business ID: 12-123-1234 Business ID Type: D-U-N-S</p>
<p>Doc Exchange</p> <p>Document Exchange Name: XOCP-spoke-de Business Protocol Binding: XOCP-1.1 Business Protocol Definition: XOCP-SPOKE End Point Type: SPOKE Confirmed Delivery: <default> Message History: <default> Retries Number of Retries: <default> Interval (Milliseconds): <default> Timeout: <default> Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol: Hash Function: Signature Algorithm:</p>	<p>Delivery Channels</p> <p>Delivery Channel Name: XOCP-spoke-dc Transport: XOCP-spoke-transport Document Exchange: XOCP-spoke-de Routing Proxy (Yes/No): No</p>
<p>Transport</p> <p>Transport Name: XOCP-spoke-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://abc.com/xocp-spoke-transport</p>	<p>Certificates</p> <p>Certificate Name: <none> Certificate Type: <none> Certificate Location: <none></p>

1 Configuration Requirements

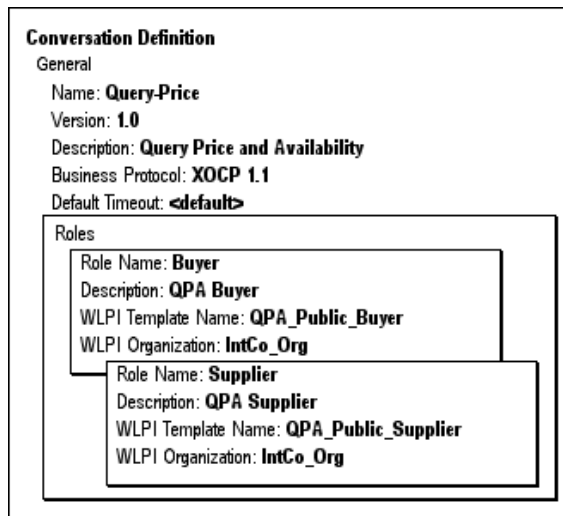
Note that the trading partner type is `Local` and the configuration includes a spoke delivery channel (`XOCP-spoke-dc`), document exchange (`XOCP-spoke-de`), and transport (`XOCP-spoke-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://abc.com/xocp-spoke-transport`).

The requirements for the XYZ, TUV, and ABC trading partner definitions in the IntCo WebLogic Integration configuration are exactly the same as those listed above, with the exception of trading partner type. Trading partner type is set to `Remote` for XYZ, TUV, and ABC in the IntCo WebLogic Integration configuration.

Conversation Definitions

The information that must be defined for the Query Price and Availability (QPA) conversation in the IntCo WebLogic Integration configuration is summarized in the following figure.

Figure 1-16 Conversation Definition for QPA

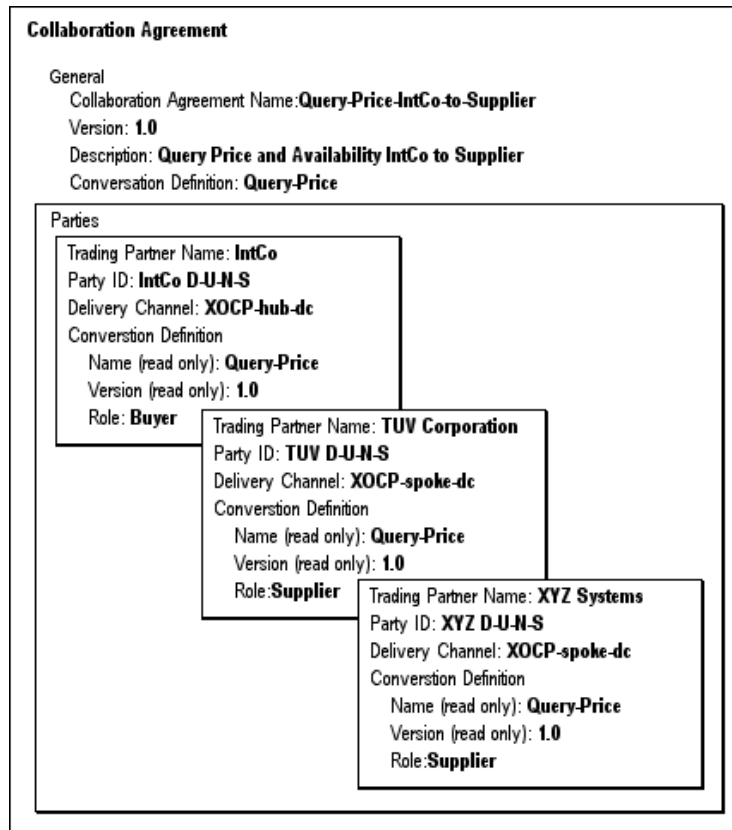


The conversation definition in the ABC, TUV, and XYZ WebLogic Integration configurations is the same, with one exception: the value of the BPM organization (listed as WLPI organization) must be changed to reflect an organization defined at each company.

Collaboration Agreements

The collaboration agreement shown in the following figure is required in the IntCo WebLogic Integration configuration. It enables the IntCo hub delivery channel to act as a proxy for the buyer role, as described in “XOCP Hub and Spoke Delivery Channels” on page 1-7.

Figure 1-17 Collaboration Agreement IntCo-to-Supplier



Note: IntCo has the option of configuring one agreement (as shown in the preceding figure) or two separate agreements (one with each supplier). If separate agreements are configured, each can be exported for use by the appropriate trading partner.

1 Configuration Requirements

Corresponding agreements are required in the TUV and XYZ configurations. The collaboration agreement for the TUV configuration is shown in the following figure.

Figure 1-18 Collaboration Agreement for TUV-IntCo

Collaboration Agreement															
General															
Collaboration Agreement Name: Query-Price-TUV-IntCo															
Version: 1.0															
Description: Query Price and Availability TUV to IntCo															
Conversation Definition: Query-Price															
Parties															
<table border="1"><tr><td>Trading Partner Name: IntCo</td></tr><tr><td>Party ID: IntCo D-U-N-S</td></tr><tr><td>Delivery Channel: XOCP-hub-dc</td></tr><tr><td>Conversation Definition</td></tr><tr><td> Name (read only): Query-Price</td></tr><tr><td> Version (read only): 1.0</td></tr><tr><td> Role: Buyer</td></tr></table>	Trading Partner Name: IntCo	Party ID: IntCo D-U-N-S	Delivery Channel: XOCP-hub-dc	Conversation Definition	Name (read only): Query-Price	Version (read only): 1.0	Role: Buyer	<table border="1"><tr><td>Trading Partner Name: TUV Corporation</td></tr><tr><td>Party ID: TUV D-U-N-S</td></tr><tr><td>Delivery Channel: XOCP-spoke-dc</td></tr><tr><td>Conversation Definition</td></tr><tr><td> Name (read only): Query-Price</td></tr><tr><td> Version (read only): 1.0</td></tr><tr><td> Role: Supplier</td></tr></table>	Trading Partner Name: TUV Corporation	Party ID: TUV D-U-N-S	Delivery Channel: XOCP-spoke-dc	Conversation Definition	Name (read only): Query-Price	Version (read only): 1.0	Role: Supplier
Trading Partner Name: IntCo															
Party ID: IntCo D-U-N-S															
Delivery Channel: XOCP-hub-dc															
Conversation Definition															
Name (read only): Query-Price															
Version (read only): 1.0															
Role: Buyer															
Trading Partner Name: TUV Corporation															
Party ID: TUV D-U-N-S															
Delivery Channel: XOCP-spoke-dc															
Conversation Definition															
Name (read only): Query-Price															
Version (read only): 1.0															
Role: Supplier															

The agreement required in the XYZ configuration is shown in the following figure.

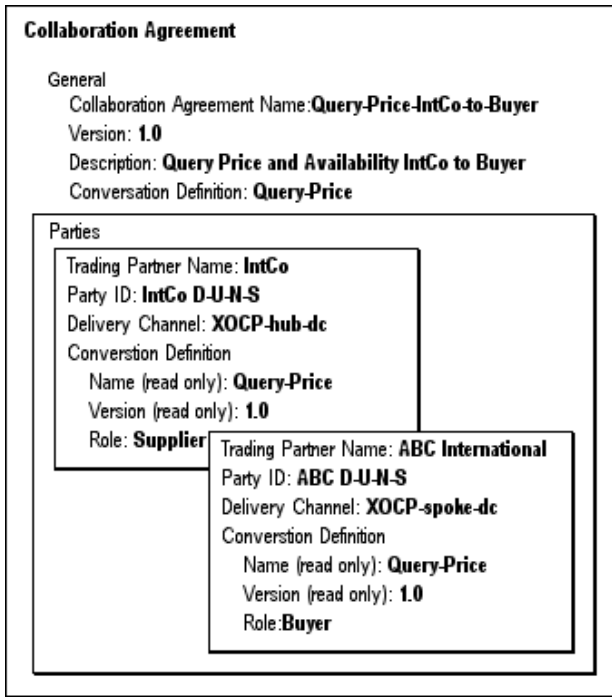
Figure 1-19 Collaboration Agreement XYZ-IntCo

Collaboration Agreement	
General	
Collaboration Agreement Name: Query-Price-XYZ-IntCo	
Version: 1.0	
Description: Query Price and Availability XYZ to IntCo	
Conversation Definition: Query-Price	
Parties	
Trading Partner Name: IntCo	Trading Partner Name: XYZ Systems
Party ID: IntCo D-U-N-S	Party ID: XYZ D-U-N-S
Delivery Channel: XOCP-hub-dc	Delivery Channel: XOCP-spoke-dc
Conversation Definition	Conversation Definition
Name (read only): Query-Price	Name (read only): Query-Price
Version (read only): 1.0	Version (read only): 1.0
Role: Buyer	Role: Supplier

1 Configuration Requirements

The collaboration agreement shown in the following figure is required in the IntCo WebLogic Integration configuration. It enables the IntCo hub delivery channel to act as a proxy for the supplier role, as described in “XOCP Hub and Spoke Delivery Channels” on page 1-7.

Figure 1-20 Collaboration Agreement IntCo-to-Buyer

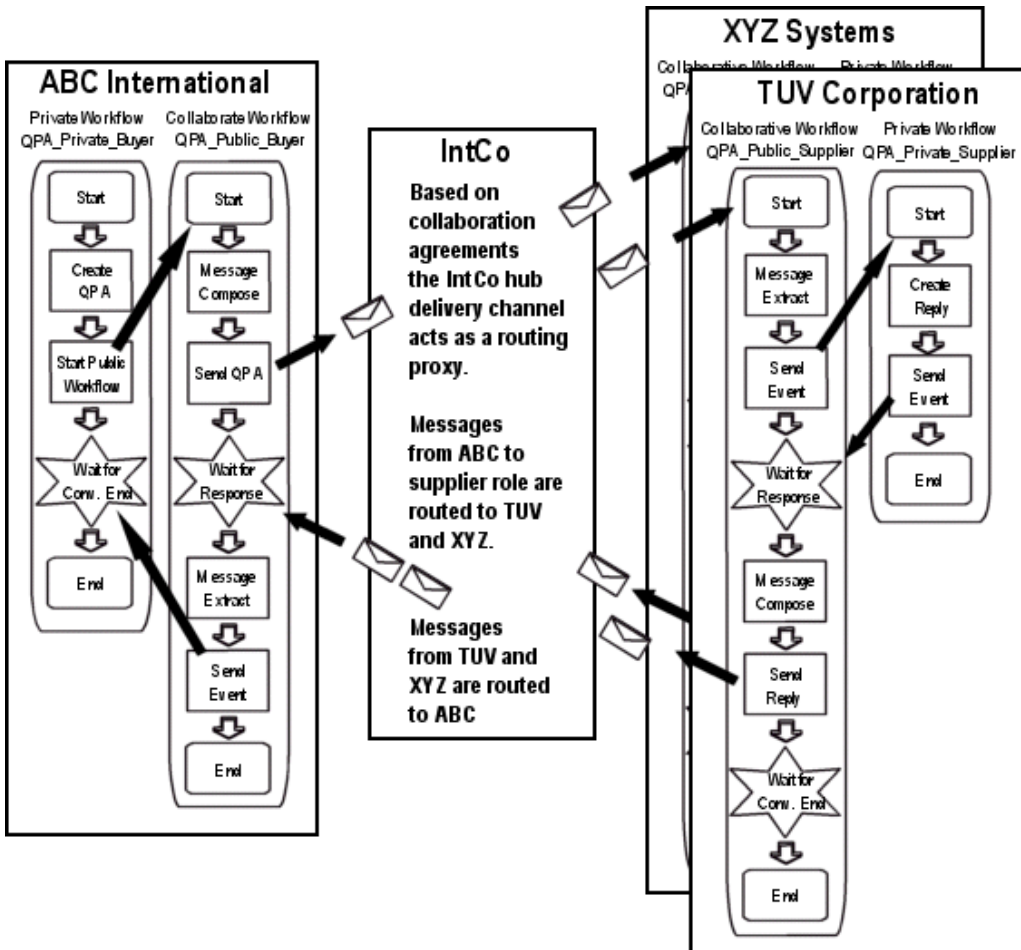


The agreement shown in this figure is also required in the ABC configuration.

How It Works

The following figure shows how IntCo mediates the exchange of Query Price and Availability messages between ABC International and the two suppliers, TUV Corporation and XYZ Systems.

Figure 1-21 QPA Collaboration Mediated by IntCo



The message sent to the supplier role by the Send QPA action of the ABC QPA_Public_Buyer workflow is forwarded to the IntCo hub delivery channel as specified by the collaboration agreement in the ABC configuration.

1 Configuration Requirements

When the IntCo hub delivery channel (XOCP-hub-dc) receives a message sent to the supplier role, collaboration agreements for the Query-Price conversation definition in which XOCP-hub-dc is assigned to the buyer role are identified. In this case, one collaboration agreement, Query-Price IntCo-to-Supplier, meets the criteria. As specified by that agreement, the message is delivered to the TUV Corporation delivery channel (XOCP-spoke-dc at <http://tuv.com/xocp-spoke-transport>) and the XYZ Systems delivery channel (XOCP-spoke-dc at <http://xyz.com/xocp-spoke-transport>).

Messages sent to the buyer role by the Send Reply action of the TUV or XYZ QPA_Public_Supplier workflow are sent to the IntCo hub delivery as specified by the collaboration agreements defined in the respective configurations.

When the IntCo hub delivery channel (XOCP-hub-dc) receives a message sent to the buyer role, collaboration agreements for the Query-Price conversation definition in which XOCP-hub-dc is assigned to the supplier role are identified. In this case, one collaboration agreement, Query-Price IntCo-to-Buyer, meets the criteria. As specified by that agreement, the message is delivered to the ABC International delivery channel (XOCP-spoke-dc at <http://abc.com/xocp-spoke-transport>).

RosettaNet Applications

In this example, the situation is the same as that described in “XOCP Peer-to-Peer Messaging” on page 1-9, where two trading partners, ABC International, a computer manufacturer, and XYZ Systems, a chip supplier, use WebLogic Integration to participate in Query Price and Availability (QPA) transactions. In this case, the trading partners use the RosettaNet protocol and employ PIP 3A2 to carry out the public processes.

Trading partners participating in Partner Interface Processes (PIPs) need to implement the public process defined by their role in the PIP, and they need to connect their internal systems, as well as their private processes and workflows, to the public process.

It is assumed that the PIP 3A2 template provided in the WebLogic Integration distribution is customized and connected to private processes that interact with internal systems as required. The collaborative workflows in this example are named PIP3A2_Product_Supplier and PIP3A2_Customer.

For general information about using the Studio (with the extended functionality provided by the B2B integration plug-in) to create collaborative workflows, see [Creating Workflows for B2B Integration](#). For information about customizing the PIP templates provided with WebLogic Integration and configuring RosettaNet security, see [Implementing RosettaNet for B2B Integration](#).

The following sections provide examples of how to:

- Configure the trading partners and their associated delivery channels for RosettaNet 2.0
- Configure a conversation definition to implement the required roles
- Associate the required trading partner delivery channels with the roles defined in the collaboration agreements

Note: Before you can use RosettaNet in a WebLogic Integration domain, you must modify the setup of the domain, as described in “[Setting Up the Environment](#)” in [Implementing RosettaNet for B2B Integration](#).

Trading Partners

Trading partner definitions for both ABC International and XYZ Systems must be provided.

Note: For simplicity, SSL or signature certificates are not used in the examples presented in this section. For information about security configuration, see [Implementing Security with B2B Integration](#).

The information that must be defined for the ABC International trading partner in the ABC configuration is summarized in the following figure.

1 Configuration Requirements

Figure 1-22 ABC International Trading Partner Definition in the ABC Configuration

ABC International WebLogic Integration Configuration
Trading Partner: ABC International

General Name: ABC International Description: Computer Manufacturer Type: Local Address: 123 ABC Street, Anytown, USA Email: admin@abc.com Phone: +1 123 456 7890 Fax: +1 123 456 8910 WLS User Name: abc Encoding: <default> State (Active, Inactive): Active	Party IDs Party ID: ABC D-U-N-S Business ID: 12-123-1234 Business ID Type: D-U-N-S
Doc Exchange Document Exchange Name: RosettaNet-de Business Protocol Binding: RosettaNet-2.0 Business Protocol Definition: RosettaNet2 Encryption Encryption Certificate: <none> Encryption Level Cipher Strength Cipher Algorithm Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol Hash Function Signature Algorithm	Delivery Channels Delivery Channel Name: RosettaNet-dc Transport: RosettaNet-transport Document Exchange: RosettaNet-de Routing Proxy (Yes/No): No
	Transport Transport Name: RosettaNet-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://abc.com/rosettanet-transport
	Certificates Certificate Name: <none> Certificate Type: <none> Certificate Location: <none>

Note that the trading partner type (listed in the General box) is `Local` and the configuration includes a single RosettaNet delivery channel (`RosettaNet-dc`), document exchange (`RosettaNet-de`), and transport (`RosettaNet-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://abc.com/rosettanet-transport`).

The ABC trading partner definition in the XYZ WebLogic Integration configuration must match the definition shown in the previous figure, with one exception: the trading partner type must be set to Remote in the XYZ WebLogic Integration configuration.

The information that must be defined for the XYZ Systems trading partner in the XYZ WebLogic Integration configuration is summarized in the following figure.

Figure 1-23 XYZ Systems Trading Partner Definition in the XYZ Configuration

XYZ Systems WebLogic Integration Configuration
Trading Partner: XYZ Systems

<p>General Name: XYZ Systems Description: Chip Supplier Type: Local Address: 456 XYZ Street, Anytown, USA Email: admin@xyz.com Phone: +1 456 789 1020 Fax: +1 456 789 1021 WLS User Name: xyz Encoding: <default> State (Active, Inactive): Active</p>	<p>Party IDs Party ID: XYZ D-U-N-S Business ID: 34-567-8910 Business ID Type: D-U-N-S</p>
<p>Doc Exchange Document Exchange Name: RosettaNet-de Business Protocol Binding: RosettaNet-2.0 Business Protocol Definition: RosettaNet2 Encryption Encryption Certificate: <none> Encryption Level Cipher Strength Cipher Algorithm Digital Signature (Nonrepudiation) Signature Certificate: <none> NonRepudiation Protocol Hash Function Signature Algorithm</p>	<p>Delivery Channels Delivery Channel Name: RosettaNet-dc Transport: RosettaNet-transport Document Exchange: RosettaNet-de Routing Proxy (Yes/No): No</p>
<p>Transport Transport Name: RosettaNet-transport Transport Protocol: HTTP Security Protocol: <none> Endpoints URI Endpoint: http://xyz.com/rosettanet-transport</p>	<p>Certificates Certificate Name: <none> Certificate Type: <none> Certificate Location: <none></p>

1 Configuration Requirements

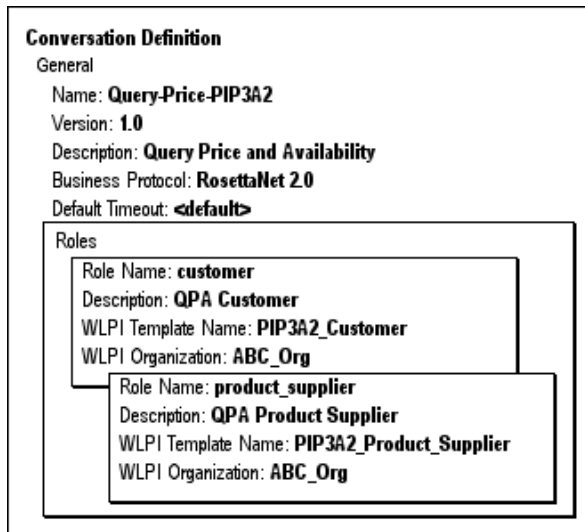
Note that the trading partner type is `Local` and the configuration includes a single RosettaNet delivery channel (`RosettaNet-dc`), document exchange (`RosettaNet-de`), and transport (`RosettaNet-transport`). The transport is associated with a URI that serves as the delivery channel endpoint (`http://xyz.com/rosettanet-transport`).

The XYZ trading partner definition in the ABC WebLogic Integration configuration must match the definition shown in the previous figure, with one exception: the trading partner type must be set to `Remote` in the ABC WebLogic Integration configuration.

Conversation Definitions

The required settings for the Query Price and Availability (QPA) conversation definition in the ABC WebLogic Integration configuration are shown in the following figure.

Figure 1-24 Conversation Definition for PIP 3A2

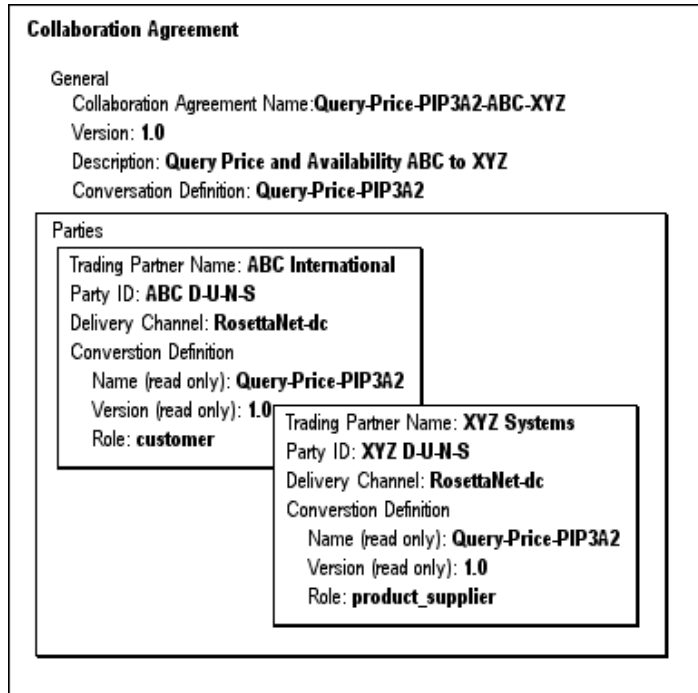


The conversation definition in the XYZ WebLogic Integration configuration is the same, with one exception: the value of the BPM organization (listed as WLPI organization) must be changed to reflect an organization defined at XYZ systems (for example, `XYZ_Org`).

Collaboration Agreements

The collaboration agreement shown in the following figure is required in both the ABC and XYZ WebLogic Integration configurations.

Figure 1-25 Collaboration Agreement Query-Price-PIP3A2-ABC-XYZ

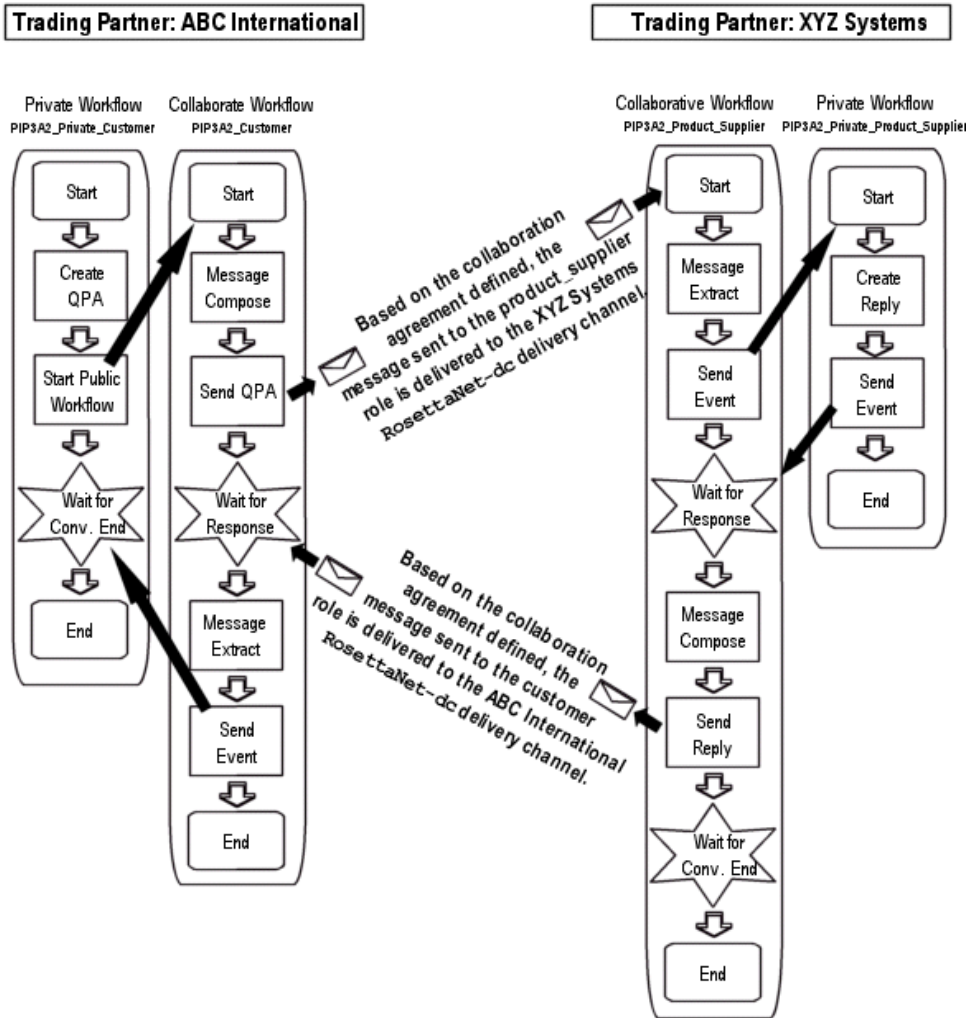


How It Works

The following figure shows how WebLogic Integration supports the exchange of Query Price and Availability messages by ABC International and XYZ Systems using RosettaNet 2.0.

1 Configuration Requirements

Figure 1-26 QPA Collaboration Overview: RosettaNet



cXML Applications

This section provides a summary of the cXML-specific requirements for configuring trading partners, conversation definitions, and collaboration agreements. For information about the architecture used to implement cXML on WebLogic Integration, cXML security, and using the cXML API, see [Implementing cXML for B2B Integration](#).

The documents exchanged in cXML are divided into three basic types: Request, Response, and Message. Within each basic type are a set of subtypes. For example, a Request might be an OrderRequest, PunchOutSetupRequest, SupplierDataRequest, SupplierListRequest, or GetPendingRequest. Each Request-Response pair constitutes a cXML transaction. For example, PunchOutSetupRequest and PunchOutSetupResponse, together, constitute the PunchOutSetup transaction.

The structure of each document adheres to the cXML DTD for a specific document type and version of cXML.

Examples summarizing the basic structure of the three major document types are shown in the following figure.

Figure 1-27 cXML Document Types

Request Document

```
<cXML version="1.1.009" payloadID="1234567.4567.5678@test.ariba.com"
timestamp="2001-03-31T18:39:09-08:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>aribaadmin@cisco.com
        </Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="DUNS">
        <Identity>012345678
        </Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>aribaadmin@cisco.com
        </Identity>
      <SharedSecret>welcome
      </SharedSecret>
      </Credential>
      <UserAgent>Ariba ORMS 6.0
      </UserAgent>
    </Sender>
  </Header>
  <Request>
    Request elements...
  </Request>
</cXML>
```

Response Document

```
<cXML version="1.1.009" payloadID="1237567.4867.5478@test.ariba.com"
timestamp="2001-03-31T18:39:09-08:00">
  <Response>
    Response elements ...
  </Response>
</cXML>
```

Message Document

```
<cXML version="1.1.009" payloadID="1234537.4527.5978@test.ariba.com"
timestamp="2001-03-31T18:39:09-08:00">
  <Header>
    Header elements ...
  </Header>
  <Message>
    Message elements...
  </Message>
</cXML>
```

When you define the trading partners and conversation definitions for cXML transactions, the values assigned to certain parameters must match specific element and attribute values that appear in the cXML root and Header elements. In addition, conversation definition names must match the cXML transaction, and the roles assigned must be `Buyer` and `Supplier`.

The following table summarizes the requirements.

Table 1-1 cXML Requirements

For a . . .	This parameter . . .	Must match . . .
Trading Partner — Supplier	Business ID Type	The value of the To Credential domain attribute. For example if <code><To><Credential domain="DUNS"></code> , then the business ID type must be set to <code>DUNS</code> .
	Business ID	The content of the To Credential Identity element. For example, if <code><To><Credential domain="DUNS"></code> <code><Identity>012345123</Identity></code> , then the business ID must be set to <code>012345678</code> .
Trading Partner — Buyer	Business ID Type	The value of the From Credential domain attribute. For example if <code><From><Credential domain="DUNS"></code> , then the business ID type must be set to <code>DUNS</code> .
	Business ID	The content of the To Credential Identity element. For example, if <code><From><Credential domain="DUNS"></code> <code><Identity>012345123</Identity></code> , then the business ID must be set to <code>012345678</code> .

1 Configuration Requirements

Table 1-1 cXML Requirements

For a . . .	This parameter . . .	Must match . . .
Conversation Definition	Name	The cXML transaction name. The name of the transaction corresponds to the name of the first child of the Request or Response element. For example, if <pre><Request> <OrderRequest> . . . </OrderRequest></pre> then the conversation definition name must be set to Order.
	Version	The value of the cXML version attribute. For example, if <pre><cXML version="1.1.009" . . . ></pre> then the conversation definition version must be set to 1.1.009.
	Roles	The roles defined must be Buyer and Supplier.

Browser Clients

In some cases, a trading partner may require little or no integration with backend systems in order to participate in a conversation. In such cases, the trading partner need not have WebLogic Integration installed. A trading partner that has WebLogic Integration can act as a host, allowing these smaller trading partners to subscribe to, and participate in, an authorized set of conversations through either a Web browser or file-sharing client.

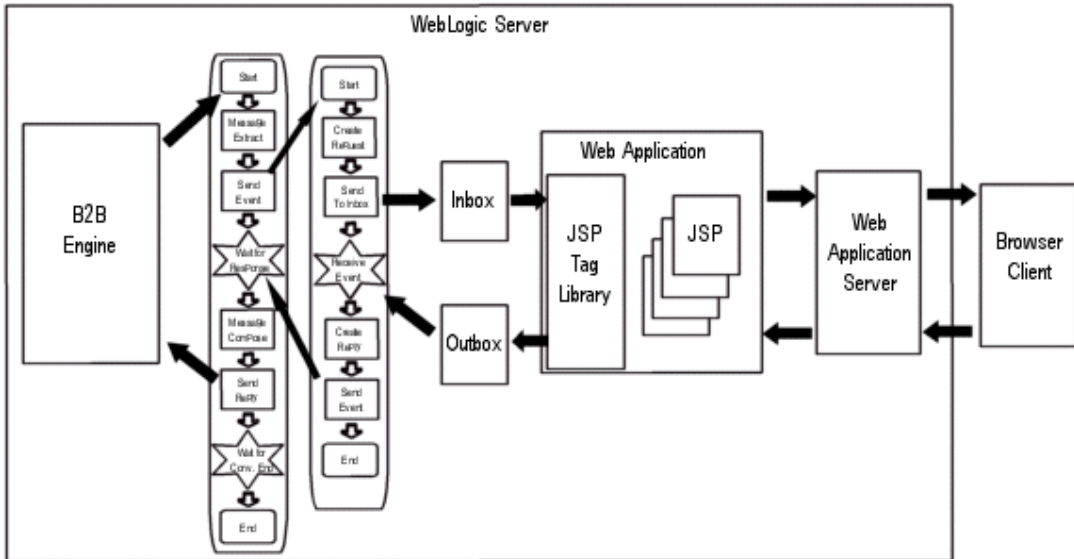
When a trading partner has no requirements for integration with backend systems, hosting the components required to allow that partner to participate as a browser client is the appropriate solution. File-sharing clients often have some requirement for backend systems integration, and usually process a higher volume of messages than browser clients.

This section discusses the requirements for supporting browser clients. The following section discusses the requirements for supporting file-sharing clients.

Before conversation messages can be presented to a browser client, processing must occur that transforms the message into a format meaningful to the browser endpoint. This processing must be hosted on behalf of the trading partner.

The following figure summarizes the elements required on the hosting trading partner.

Figure 1-28 Browser Client Host



A Web application, which includes the required Java Server Pages (JSPs), servlets, style sheets, static HTML pages, JSP tag library, scripts, and applets, provides the interface to the browser client.

Incoming and outgoing mailboxes, which provide reliable, secure storage for browser client messages, are created via the Web application using tags from the JSP tag library. The JSP tag library is included in the WebLogic Integration distribution. It provides the interface to mailboxes, supports the creation and removal of mailboxes, and allows Browser clients to administer stored messages.

CreatemboxTag is used to create mailboxes. The mailboxes must be named as follows:

- For the incoming mailbox use: *trading_partner_name_Inbox*
- For the outgoing mailbox use: *trading_partner_name_Outbox*

Workflows provide the interface between the mailboxes and the B2B engine. Although the workflow that interacts with the mailboxes can be the collaborative workflow that implements the trading partner role, for the purposes of this discussion it is assumed to be a private workflow that starts, or is started by, the collaborative workflow.

1 Configuration Requirements

Appropriately formatted XML messages are exchanged between:

- The Web application and the mailboxes
- The private workflow and the mailboxes

The messages are exchanged as follows:

- *Messages from the Web application to the outgoing mailbox*
XML messages from the Web application are sent to the outgoing mailbox for the browser client using the JSP tag library `SendMsgTag`. If this message is a response to a message that was sent by the private workflow, the workflow instance ID is embedded in the message.
- *Messages from the outgoing mailbox to the private workflow*
Upon the arrival of a message in the outgoing mailbox, the mailbox listen method is invoked automatically and posts an XML event to the internal event JMS topic for the mailbox. The private workflow event subscribed to the topic for that mailbox is triggered, and processes or forwards the message as required.
- *Messages from the private workflow to the incoming mailbox*
A private workflow business operation posts the XML messages to the incoming mailbox. The workflow instance ID is embedded in the message.
- *Message from the incoming mailbox to the Web application*
The browser client can use the JSP tag library `ChecknewmsgTag` or `CheckallmsgTag` to retrieve messages from the incoming mailbox.

The WebLogic Integration distribution includes a browser client sample. Core components of the sample Web application and workflows provided can be customized, or reused without change, to implement support for your browser clients. For information about customizing components of the sample Web application, see [“Trading Partner Lightweight Client Sample”](#) in *Running the B2B Integration Samples*.

Once you have developed the JSP pages required, and modified the other components as described, the components can be packaged in a Web Application Archive (WAR) file and deployed as required on WebLogic Server.

The following example summarizes the information that must be defined to support a browser client.

Hosting a Browser Client

In this example, the situation is the same as that described in “RosettaNet Applications” on page 1-30: two trading partners, ABC International, a computer manufacturer, and XYZ Systems, a chip supplier, participate in Query Price and Availability (QPA) transactions. In this case, ABC International hosts the components required to allow XYZ Systems to participate as a browser client.

The collaborative and private customer workflows, `PIP3A2_Customer` and `PIP3A2_Private_Customer`, are the same as the workflows used in the RosettaNet example. The collaborative and private workflows, `PIP3A2_Product_Supplier` and `PIP3A2_Private_Web_Product_Supplier`, for the product supplier role are assumed to have been customized to support the XYZ Systems as a browser client.

All workflows are hosted on the ABC International system.

The following elements are configured on the host system, ABC International:

■ *Trading Partners*

The trading partner definition for ABC International is the same as the definition shown in Figure 1-22.

The trading partner definition for XYZ Systems is the same as the definition shown in Figure 1-23, with the following exceptions:

- The definition resides in the ABC configuration.
- The transport URI endpoint is changed to `http://abc.com/rosettanel-client-transport`.
- The trading partner name is changed to `XYZ_Systems`. (The space is replaced by an underscore because spaces are not used in mailbox names, and the trading partner name is part of the mailbox name.)

■ *Conversation Definition*

The conversation definition is the same as the definition shown in Figure 1-24.

■ *Collaboration Agreements*

The collaboration agreement is the same as the agreement shown in Figure 1-25, except that the trading partner name for XYZ Systems is modified (the space is replaced by an underscore).

1 Configuration Requirements

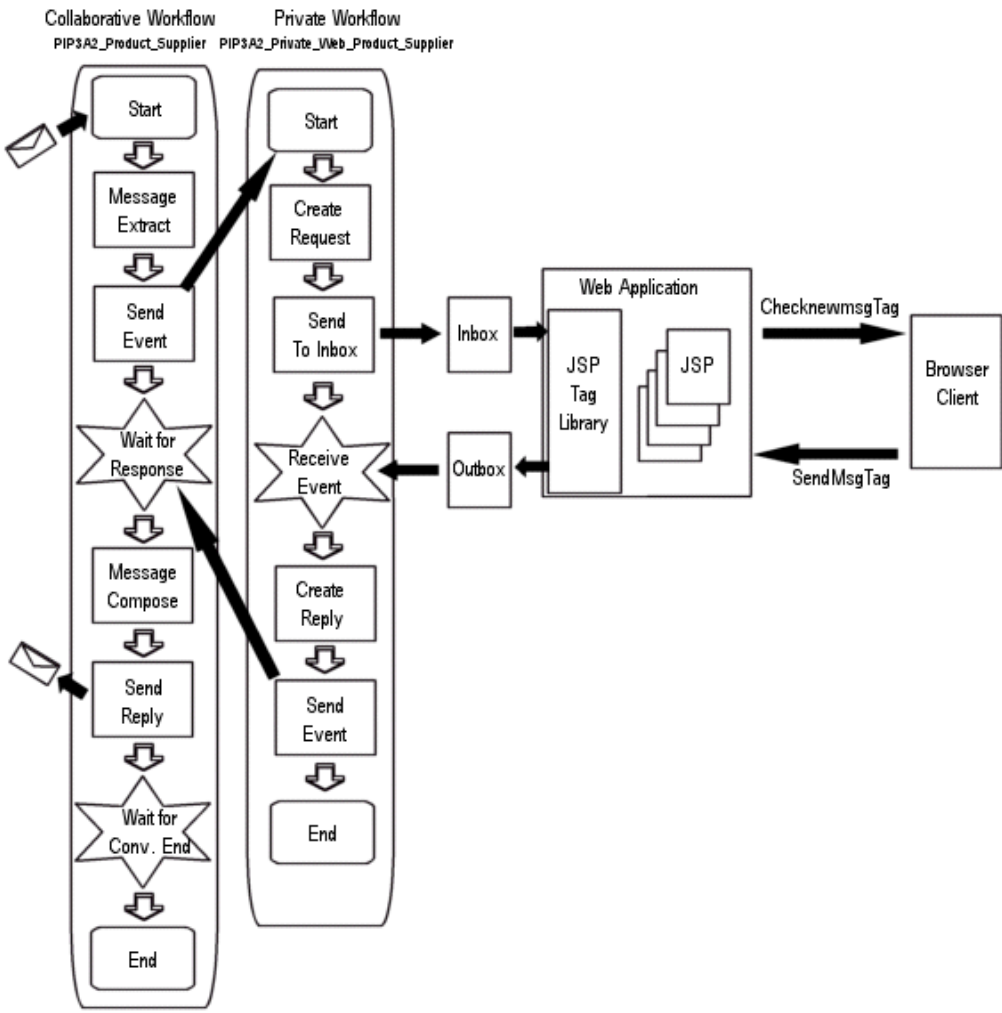
The exchange of messages between the PIP3A2_Customer and PIP3A2_Product_Supplier workflows is essentially the same as the exchange shown in Figure 1-26. The only difference is that in this example, the exchange takes place within the same WebLogic Integration instance.

The processing required to support XYZ Systems as a browser client occurs in the Web application and in the PIP3A2_Private_Web_Product_Supplier workflow. This workflow now:

- Transforms the message received from the Send Event task node of PIP3A2_Public_Product_Supplier into an XML message appropriately formatted for delivery to the Web application
- Posts the XML message to XYZ_Systems_Inbox
- Waits for a response to be posted to XYZ_Systems_Outbox
- Transforms the response into a message appropriately formatted for receipt by the Wait for Response event node of the PIP3A2_Public_Product_Supplier

The following figure provides a summary of the actions performed between the time at which a message is received on the Start node and the time at which the Send Reply task of the PIP3A2_Private_Web_Product_Supplier workflow is executed.

Figure 1-29 QPA Collaboration Overview: Browser Client



File-Sharing Clients

As described in the preceding section, a trading partner that has WebLogic Integration can act as a host for other trading partners that require little or no integration with backend systems.

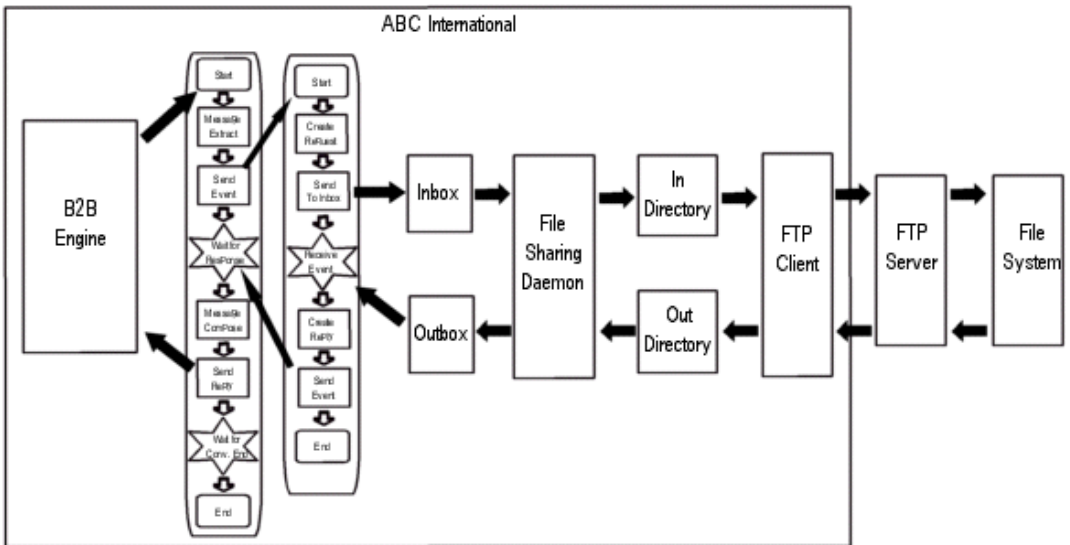
When a trading partner has minimal requirements for integration with backend systems, hosting the components required to allow that partner to participate as a file-sharing client is usually the appropriate solution. File-sharing clients often process a higher volume of messages than browser clients.

In the preceding section, the requirements for supporting browser clients were discussed. This section discusses the requirements for supporting file-sharing clients.

Before conversation messages can be presented to a file-sharing client, processing must occur that transforms the message into a format meaningful to the file-sharing endpoint. This processing must be hosted on behalf of the trading partner.

The following figure summarizes the elements required on the hosting trading partner.

Figure 1-30 File-Sharing Host



Like the browser client case, workflows provide the interface to incoming and outgoing mailboxes which are created using the JSP tag library `CreateMailboxTag`. In the file-sharing client case, the host system administrator usually creates these mailboxes on behalf of the file-sharing client. The mailboxes must be named as follows:

- For the incoming mailbox use: `trading_partner_name_Inbox`
- For the outgoing mailbox use: `trading_partner_name_Outbox`

WebLogic Integration provides the file-sharing daemon. The file-sharing daemon synchronizes files between the incoming and outgoing directories on the local file system with the incoming and outgoing mailboxes, as follows:

- The daemon polls the incoming mailbox at specified intervals. Newly arrived files are copied into the corresponding incoming directory.
- The daemon polls the outgoing directory on the file system at specified intervals. Newly arrived files are copied into the corresponding outgoing mailbox.

An FTP client, supplied by either a customer or a third party, serves as the interface to the incoming and outgoing directories on the file system. The mechanism for transferring the message files from the incoming and outgoing directories to the file system at the file-sharing client location, and the processing required to send or reply to the messages, is implemented by the file-sharing client.

The following example summarizes the information that must be defined to support a file-sharing client.

Hosting a File-Sharing Client

In this example, the situation is the same as that described in “Browser Clients” on page 1-40: two trading partners, ABC International, a computer manufacturer, and XYZ Systems, a chip supplier, plan to participate in Query Price and Availability (QPA) transactions. In this case, ABC International has agreed to host the components required to allow XYZ Systems to participate as a file-sharing client.

The collaborative and private customer workflows, `PIP3A2__Customer` and `PIP3A2__Private_Customer`, are the same as the workflows used in the browser client example. The collaborative and private workflows,

1 Configuration Requirements

PIP3A2_Product_Supplier and PIP3A2_Private_FTP_Product_Supplier, for the product supplier role are assumed to have been customized to support XYZ Systems as a file-sharing client.

All workflows are hosted on the ABC International system.

ABC International must modify the `config.xml` file for the domain and edit the configuration file for the file-sharing daemon. For the required modifications, see “Configuring a Lightweight Client” in “[Trading Partner Lightweight Client Sample](#)” in *Running the B2B Integration Samples*.

The following elements are configured on the host system, ABC International:

■ *Trading Partners*

The trading partner definition for ABC International is the same as the definition shown in Figure 1-22.

The trading partner definition for XYZ Systems is the same as the definition shown in Figure 1-23, with the following exceptions:

- The definition resides in the ABC configuration.
- The transport URI endpoint is changed to `http://abc.com/rosettanet-client-transport`.
- The trading partner name is changed to `XYZ_Systems`.
(The space is replaced by an underscore because spaces are not used in mailbox names, and the trading partner name is part of the mailbox name.)

■ *Conversation Definition*

The conversation definition is the same as the definition shown in Figure 1-24.

■ *Collaboration Agreements*

The collaboration agreement is the same as the agreement shown in Figure 1-25, except that the trading partner name for XYZ Systems is modified (a space is replaced by an underscore).

The exchange of messages between the `PIP3A2__Customer` and `PIP3A2__Product_Supplier` workflows is essentially the same as the exchange shown in Figure 1-26. The only difference is that now, this exchange takes place within the same WebLogic Integration instance.

The processing required to support XYZ Systems as a file-sharing client occurs at the file-sharing location and in the `PIP3A2_Private_FTP_Product_Supplier` workflow. This workflow now:

- Transforms the message received from the Send Event task node of PIP3A2_Public_Product_Supplier into a message appropriately formatted for delivery to the file-sharing client
- Posts the message to XYZ_Systems_Inbox
- Waits for a response to be posted to XYZ_Systems_Outbox
- Transforms the response into a message appropriately formatted for receipt by the Wait for Response event node of the PIP3A2_Public_Product_Supplier

1 *Configuration Requirements*

2 Basic Configuration Tasks

This section provides an overview of the tasks and procedures required to configure BEA WebLogic Integration for trading exchange, supply chain management, and collaborative commerce applications. It includes the following topics:

- Overview of WebLogic Integration B2B Console
- Getting Help
- Configuring the B2B Engine
- Configuring Trading Partners
- Configuring Conversation Definitions
- Configuring Collaboration Agreements

The detailed information required to perform the tasks outlined in this section is provided in the B2B Console online help. For information about accessing the help, see “Getting Help” on page 2-7.

For example configuration scenarios, see Chapter 1, “Configuration Requirements.”

Advanced features, such as configuring extended properties for a trading partner, configuring logic plug-ins, and using XPath expressions to control the flow of XOCB business messages, are discussed in Chapter 3, “Advanced Configuration Tasks.”

Overview of WebLogic Integration B2B Console

The WebLogic Integration B2B Console is used to:

- Configure B2B engine preferences, security, and proxy settings
- Configure trading partners, conversation definitions, collaboration agreements, business protocol definitions, and logic plug-ins
- Export and import the entire repository or a selected subset of its elements
- Monitor the B2B engine, trading partner sessions, conversations, and collaboration agreements

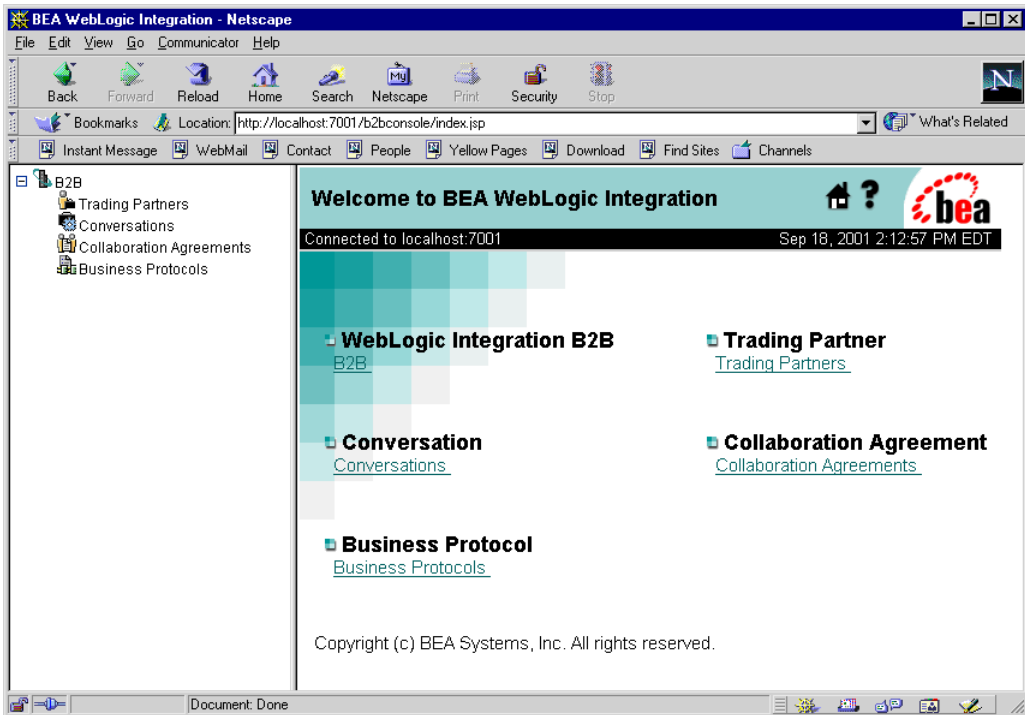
Once you have started WebLogic Integration as described in “[Getting Started](#)” in *Starting, Stopping, and Customizing BEA WebLogic Integration*, you can access the B2B Console by navigating to the following URL:

```
http://host:7001/b2bconsole
```

Here *host* is the computer name or IP address of the system that is running WebLogic Integration, and *7001* is the WebLogic Server listen port configured for the domain. Specify `localhost` or `127.0.0.1` if the server is running on the local computer.

The B2B Console home page is displayed as shown in the following figure.

Figure 2-1 WebLogic Integration B2B Console



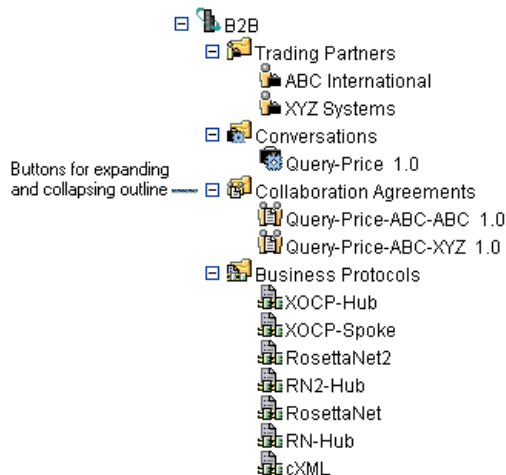
Like the navigation tree in the WebLogic Server Administration Console, the navigation tree in the left pane of the B2B Console contains a list of B2B Console pages with links that let you open those pages in the right pane.

The items displayed in the navigation tree and the tabs available on specific B2B Console pages are controlled by the following options on the Preferences tab:

- *Hide advanced configuration controls*
By default, the Hide advanced configuration controls option is selected on the B2B Preferences tab. When this option is selected, logic plug-ins are not displayed in the navigation tree, and the Advanced tab is not displayed when a trading partner is selected. Advanced features are discussed in Chapter 3, “Advanced Configuration Tasks.”
- *Display entities on the navigation tree*
By default, the Display entities on the navigation tree option is not selected on the B2B Preferences tab. When this option is selected, the entities defined for your application (including trading partners, conversation definitions, collaboration agreements, business protocols, and logic plug-ins) are available for selection from the navigation tree, as shown in the following figure.

For information about these options, see “Setting Preferences” in the [Online Help for the WebLogic Integration B2B Console](#).

Figure 2-2 Navigation Tree



The B2B Console navigation tree and the pages available from it are summarized in the following figures.

Figure 2-3 WebLogic Integration B2B Console

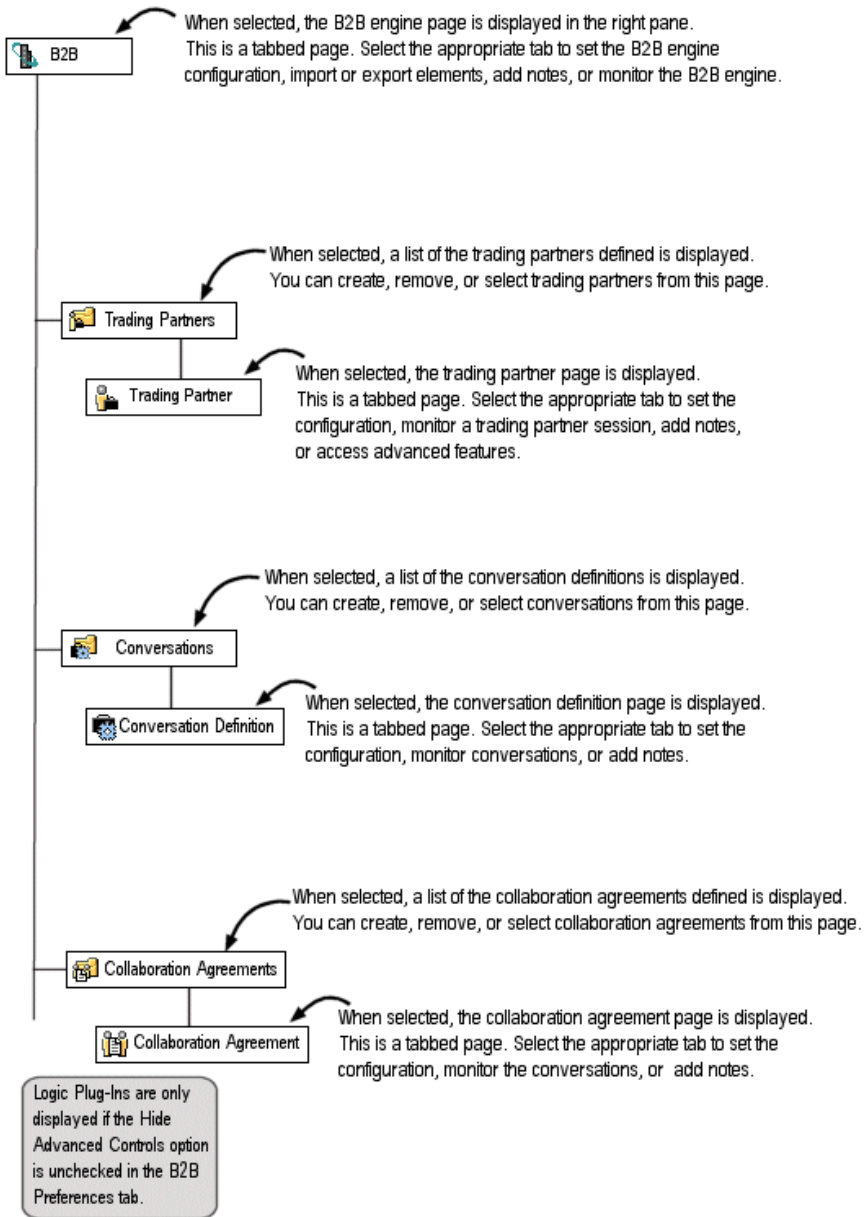
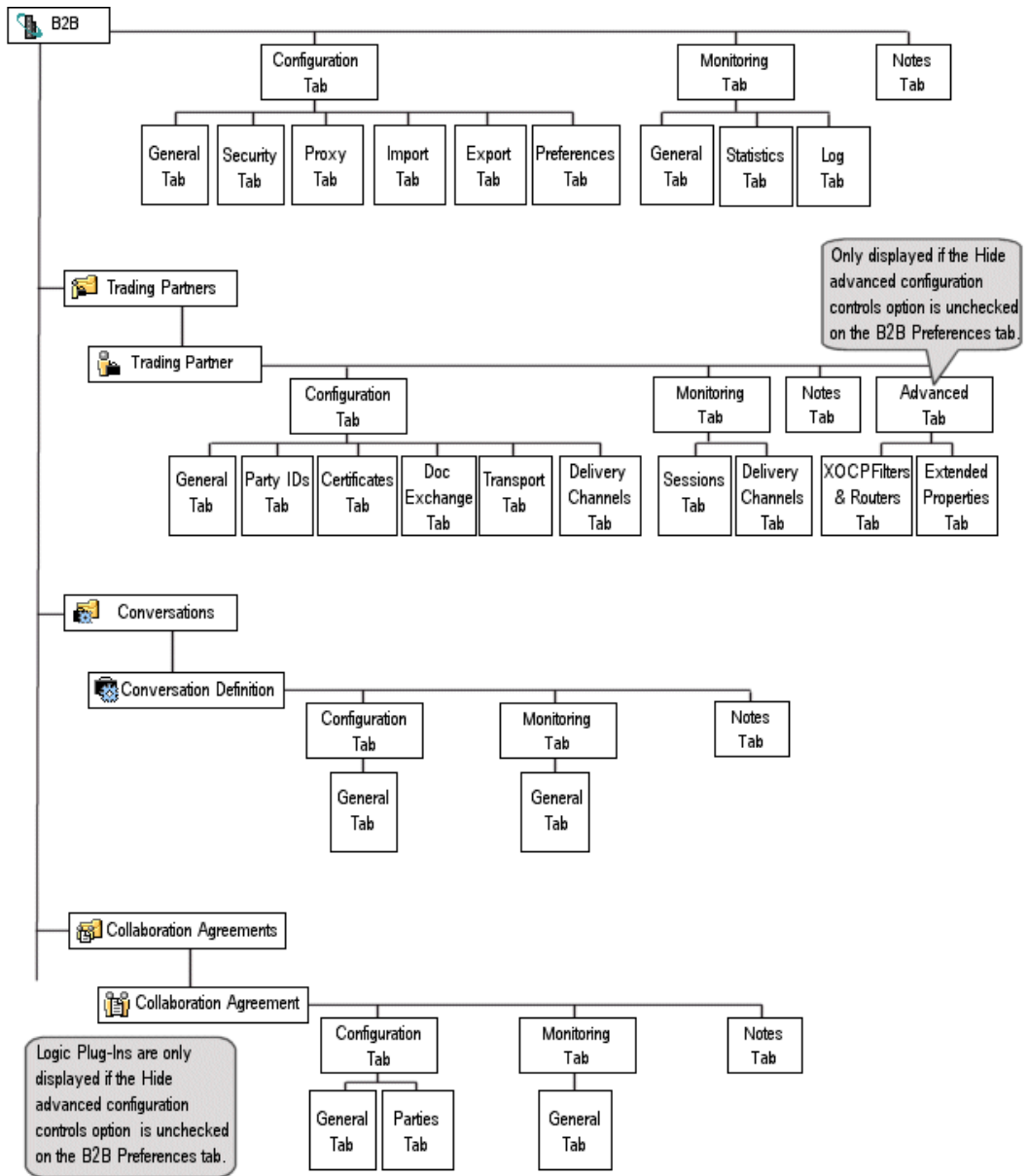





Figure 2-4 WebLogic Integration B2B Console




Getting Help

You can get context-sensitive help at any time by clicking the question mark in the upper right corner of any B2B Console page, as shown in the following figure.


Figure 2-5 Getting Help

B2B> Trading Partners   

Connected to localhost:7001 Sep 19, 2001 11:27:44 AM EDT

 [Create a new Trading Partner...](#)

Previous 5 | Next 5 | First | Last | [Refresh](#)

	Trading Partners	Collaboration Agreements	Type
<input type="checkbox"/>	ABC International	2	LOCAL
<input type="checkbox"/>	XYZ Systems	1	REMOTE

Previous 5 | Next 5 | First | Last | [Refresh](#)

The [Online Help for the WebLogic Integration B2B Console](#) document is also included in the WebLogic Integration 2.1 product documentation.

Configuring the B2B Engine

When you select B2B from the navigation tree, the B2B engine page is displayed. This page, in turn, displays a set of three tabs (Configuration, Monitoring, and Notes), each of which offers a set of nested tabs. These tabs enable you to do the following tasks:

- Configuration tab—View or modify the B2B engine configuration in various ways, including by importing or exporting configured trading partners, conversation definitions, collaboration agreements, logic plug-ins, business protocols, and repository data
- Monitoring tab—Track the progress of the B2B engine, and shut down or restart it
- Notes—Add information specific to your B2B engine

When you first access the B2B engine page, the Configuration tab is displayed with the nested General tab selected, as shown in the following figure.

Figure 2-6 Configuration Tab in the B2B Engine Page

The screenshot displays the Configuration tab interface for the B2B engine. At the top, there are three main tabs: Configuration (selected), Monitoring, and Notes. Under the Configuration tab, there are six sub-tabs: General (selected), Security, Proxy, Import, Export, and Preferences. The main content area shows the following configuration options:

- WLI B2B Name:** B2B
- Description:** [Empty text field]
- Large Message Support:**
 - Use Large Message Support
 - Location:** [Empty text field]
 - Minimum Size:** 0 [Empty text field] KB
- Buttons:** Apply, Reset

Using the nested tabs accessed from the Configuration tab, you can set parameters for the following items:

- **General attributes**—This tab allows you add a description to the B2B engine, and to set parameters related to large message support.
- **Security**—This tab allows you to define basic security for the B2B engine.
- **Proxy**—This tab allows you to define a proxy server.
- **Import**—This tab allows you to import B2B configuration data from an XML file.
- **Export**—This tab allows you to specify the parameters of an export.
- **Preferences**—This tab allows you to specify the default character set for data entered through the B2B Console. For additional information about using alternate character sets, see “Using an Alternate Character Set” in [“Customizing WebLogic Integration”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

The detailed information required for these configuration tasks is provided in the B2B Console online help. See “Getting Help” on page 2-7.

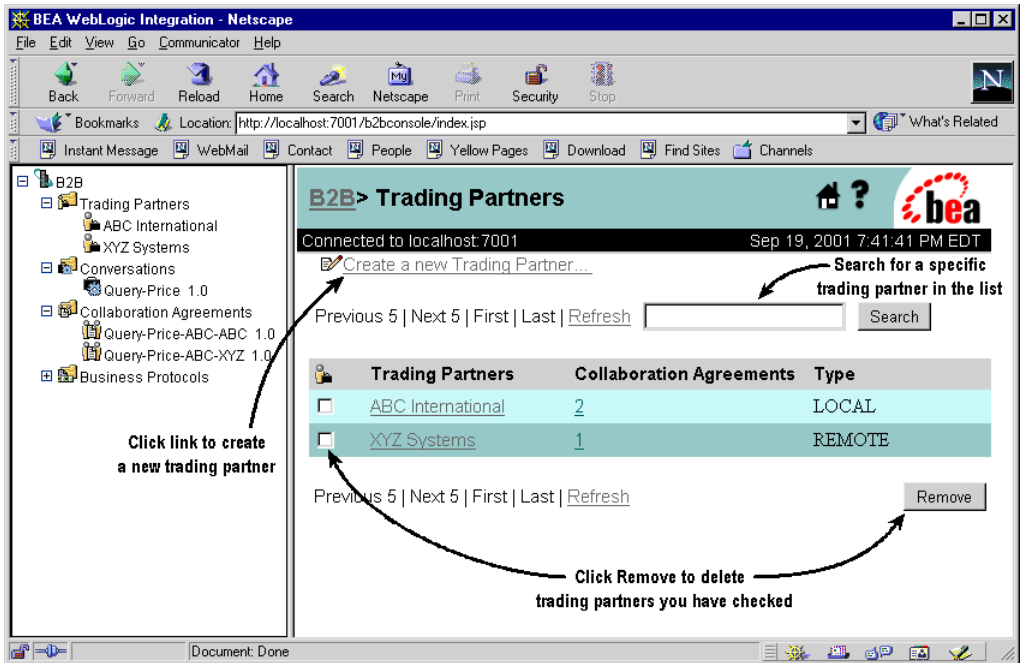
Note: Certain constraints apply to the modification of some B2B engine parameters. See Appendix A, “Update Considerations,” before making changes to your configuration.

Configuring Trading Partners

If the individual trading partner that you want to configure is listed in the navigation tree, you can invoke the configuration page for it by selecting its name in the tree. If a list of available trading partners is not displayed in the navigation tree, select the Trading Partners node.

When you select the Trading Partners node from the navigation tree, the Trading Partners page is displayed in the right pane. This page shows a list of the currently defined trading partners, along with options for selecting, searching for, or removing an existing trading partner, and creating a new one.

Figure 2-7 Trading Partners Page



When you select either Create a new Trading Partner or an existing trading partner, a configuration page for the specified trading partner is displayed. The first time you access this page for the specified trading partner, the Configuration tab is displayed with the nested General tab selected.

Figure 2-8 Individual Trading Partner Page

The screenshot shows a web interface for configuring a trading partner. At the top, there are several tabs: Configuration, Monitoring, Notes, and Advanced. The 'Configuration' tab is active, and it contains sub-tabs: General, Party IDs, Certificates, Doc Exchange, Transport, and Delivery Channels. The 'General' sub-tab is selected, displaying the following fields:

- Name: ABC International
- Description: Computer Manufacturer
- Type: LOCAL (dropdown menu)
- Address: 123 ABC Street, Anytown, USA
- Email: admin@abc.com
- Phone: +1 123 456 7890
- Fax: +1 123 456 8910
- WLS User Name: abc
- Encoding: (empty field)
- State:
 - Active
 - Inactive

At the bottom right of the form are 'Apply' and 'Reset' buttons. A callout box with an arrow pointing to the 'Advanced' tab contains the text: "The Advanced tab is only displayed if the Hide advanced configuration controls option is unchecked on the B2B Preferences tab".

The tabs on this page allow you to:

- View or modify the trading partner configuration
- Monitor the trading partner sessions and delivery channels
- Add notes to the trading partner configuration

- Perform advanced configuration tasks:
 - Associate XPath expressions with the trading partner to control the flow of business messages
 - Define extended properties required by your application or used in XPath expressions

Note: As noted in Figure 2-4, the Advanced tab is displayed only if the Hide advanced configuration controls option is unselected on the B2B Preferences tab. For a discussion of these advanced features, see Chapter 3, “Advanced Configuration Tasks.”

Configuring a trading partner involves setting parameters for the following items:

- Basic identifying information
- Trading partner party IDs
- Trading partner security certificates
- Trading partner document exchanges
- Trading partner transports
- Trading partner delivery channels

The detailed information required for these configuration tasks is provided in the B2B Console online help. See “Getting Help” on page 2-7.

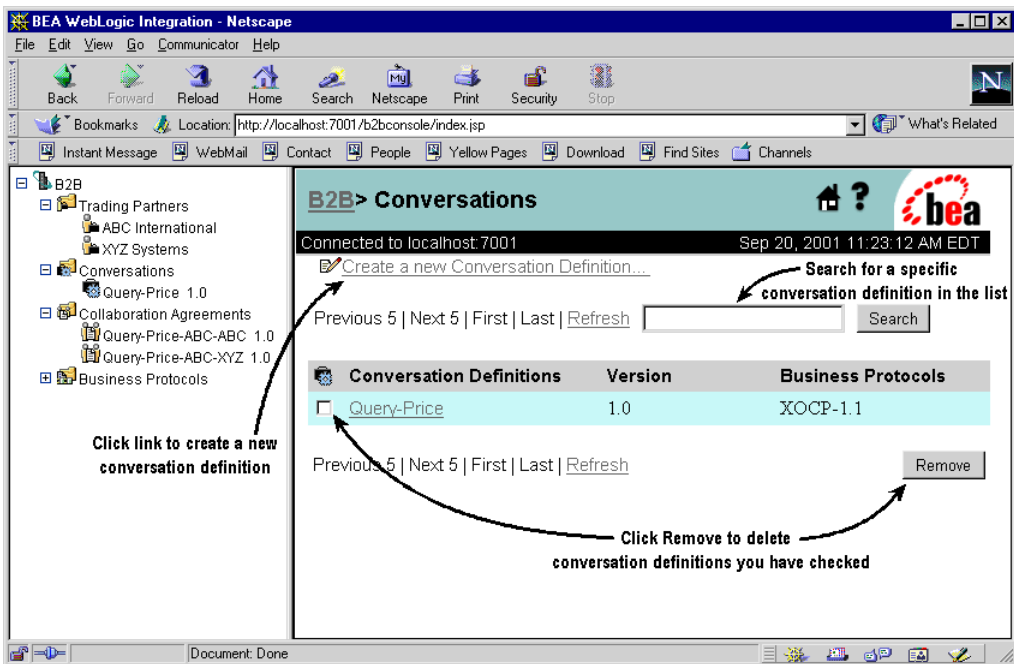
Note: Certain constraints apply to the modification of some trading partner parameters. See Appendix A, “Update Considerations,” before making changes to your configuration.

Configuring Conversation Definitions

If the individual conversation definition that you want to configure is listed in the navigation tree, you can invoke the configuration page for it by selecting its name in the tree. If a list of available conversation definitions is not displayed in the navigation tree, select the Conversation node.

When you select the Conversation node from the navigation tree, the Conversations page is displayed in the right pane. This page shows a list of the currently defined conversations, along with options for selecting, searching for, or removing an existing conversation, and creating a new one.

Figure 2-9 Conversations Page



When you select either Create a new Conversation Definition or an existing conversation definition, a configuration page for the specified conversation definition is displayed. The first time you access this page for a specified conversation definition, the Configuration tab is displayed with the nested General tab selected.

Figure 2-10 Individual Conversation Definition Page

The screenshot shows a web interface for configuring a conversation definition. At the top, there are three tabs: "Configuration" (selected), "Monitoring", and "Notes". Under the "Configuration" tab, the "General" sub-tab is active. The main form area contains the following fields:

- Name:** Query-Price
- Version:** 1.0
- Description:** Query Price and Availability
- Business Protocol:** XOCP-1.1 (dropdown menu)
- Default Timeout:** 0 ms

Below these fields is a section titled "Roles" with the following fields:

- Name:** Buyer
- Description:** QPA Buyer
- WLPI Template Name:** QPA_Public_Buyer
- WLPI Organization:** ABC_Org

At the bottom of the Roles section is a sub-section titled "Available Roles" containing a list box with "Buyer" and "Supplier". Below the list box are "Set" and "Remove" buttons. At the bottom right of the main form area are "Apply" and "Reset" buttons.

The tabs on this page allow you to:

- View or modify the conversation definition configuration
- Monitor conversations
- Add notes to the conversation definition configuration

Configuring a conversation definition involves the following tasks:

- Configuring basic identifying information
- Defining the conversation roles, and assigning a business process management (BPM) workflow template and organization to the role

The detailed information required for these configuration tasks is provided in the B2B Console online help. See “Getting Help” on page 2-7.

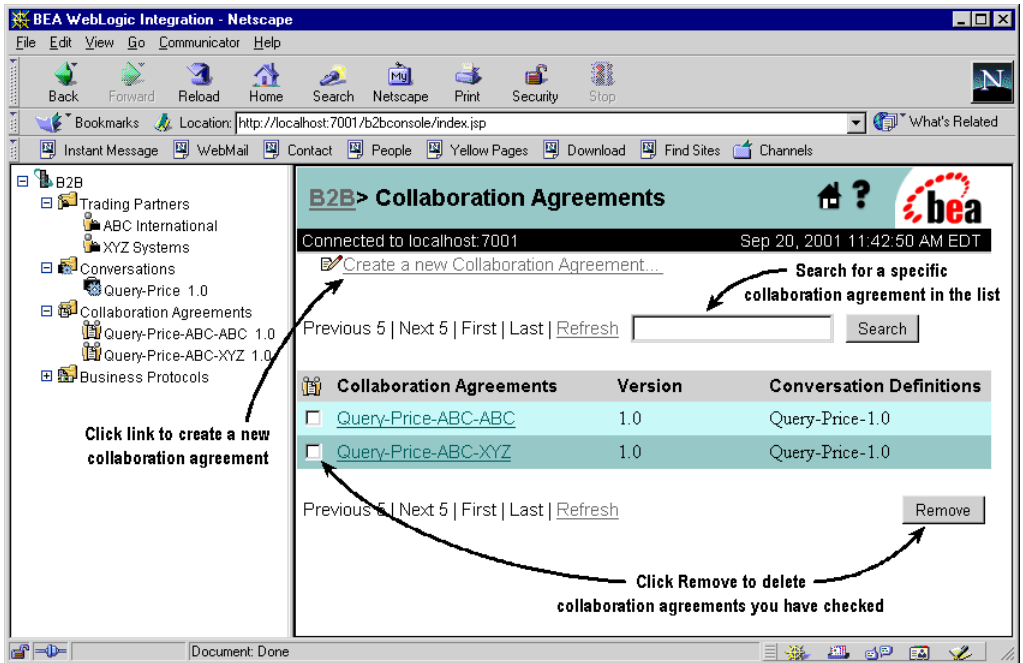
Note: Certain constraints apply to the modification of some conversation definition parameters. See Appendix A, “Update Considerations,” before making changes to your configuration.

Configuring Collaboration Agreements

If the individual collaboration agreement that you want to configure is listed in the navigation tree, you can invoke the configuration page for it by selecting its name in the tree. If a list of available collaboration agreements is not displayed in the navigation tree, select the Collaboration Agreements node.

When you select the Collaboration Agreements node from the navigation tree, the Collaboration Agreement page is displayed in the right pane. This page shows a list of the currently defined configuration agreements, along with options for selecting, searching for, or removing an existing collaboration agreement, and creating a new one.

Figure 2-11 Collaboration Agreements Page



When you select either Create a new Collaboration Agreement or an existing collaboration agreement, a configuration page for the specified configuration agreement is displayed. The first time you access this page for the specified trading partner, the Configuration tab is displayed with the nested General tab selected.

Figure 2-12 Individual Collaboration Agreement Page

The screenshot shows a web interface for configuring a collaboration agreement. At the top, there are three tabs: 'Configuration' (selected), 'Monitoring', and 'Notes'. Under the 'Configuration' tab, there are two sub-tabs: 'General' (selected) and 'Parties'. The 'General' sub-tab contains the following fields:

- Collaboration Agreement Name:** Query-Price-ABC-ABC
- Version:** 1.0
- Description:** Query Price and Availability ABC Hub to
- Conversation Definition:** Query-Price&version=1.0

At the bottom right of the configuration area, there are two buttons: 'Apply' and 'Reset'.

The tabs on this page allow you to:

- View or modify the collaboration agreement configuration
- Add notes to the collaboration agreement configuration

Configuring a collaboration agreement involves defining the following:

- Basic identifying information
- Collaboration agreement parties

The detailed information required for these configuration tasks is provided in the B2B Console online help. See “Getting Help” on page 2-7.

Note: Certain constraints apply to the modification of some collaboration agreement parameters. See Appendix A, “Update Considerations,” before making changes to your configuration.

3 Advanced Configuration Tasks

This section introduces the advanced features of the B2B integration functionality provided by WebLogic Integration. A roadmap to detailed information about these features is also provided. The section includes the following topics:

- Overview of Advanced Features
- XPath Expressions in Routing and Filtering
- Configuring Router and Filter Expressions
- Custom Logic Plug-Ins
- Trading Partner Extended Properties

Overview of Advanced Features

Logic plug-ins are Java classes that are invoked when WebLogic Integration is started. At run time, they intercept, process, and output business messages. The advanced features associated with logic plug-ins include:

- *Routing and filtering of business messages*

The built-in XOCB router and XOCB filter logic plug-ins support customer-defined XPath router and filter expressions. Based on the XPath router and filter expressions defined, you can control the flow of XOCB business messages exchanged among trading partners as follows:

- The XPath router logic plug-in can modify the list of recipients for an XOCB business message.
- The XPath filter logic plug-in can determine whether an XOCB business message is sent to a trading partner.

Note: The use of XPath router and filter expressions for routing messages to trading partners is a special feature of the XOCB business protocol.

- *Custom logic plug-ins*

By default, a chain of built-in, business protocol-specific logic plug-ins is associated with each business protocol routing and filtering function. Custom logic plug-ins can be defined and inserted, where required, in either the routing or filtering chain. These custom plug-ins can perform a wide range of services besides routing and filtering.

In addition, the B2B integration functionality provided by WebLogic Integration supports the specification of extended properties for trading partners. These extended properties can be used by:

- XPath routing and filtering expressions
- Custom logic plug-ins
- External applications

XPath Expressions in Routing and Filtering

The XPath expressions used by the XOCP router and XOCP filter logic plug-ins to control the flow of business messages fall into three categories:

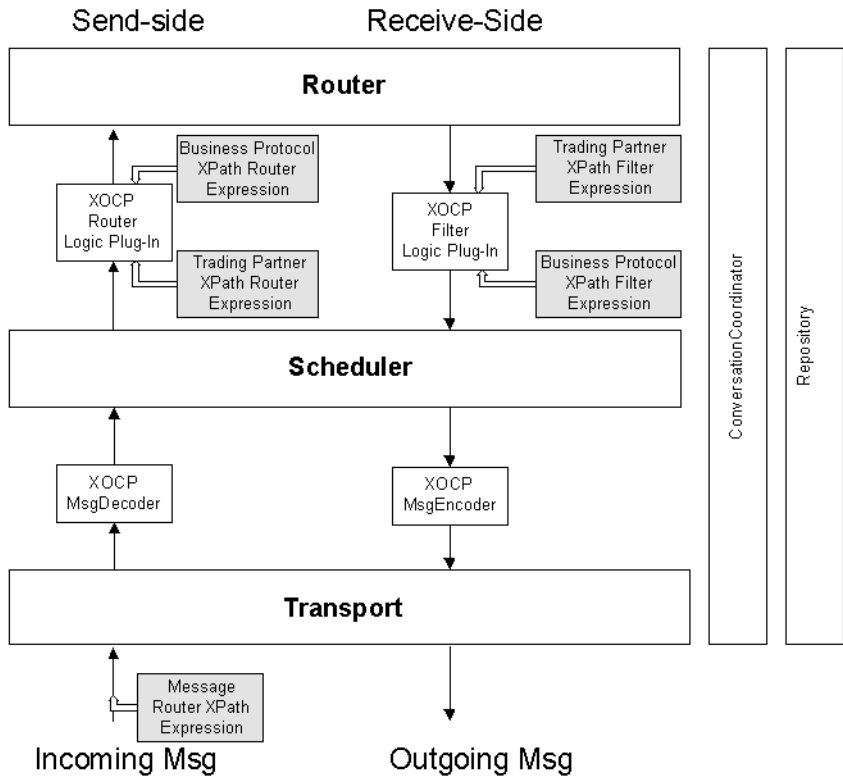
- *XPath Expressions added when the business message is assembled*
Message XPath router expressions can be associated directly with a message. When an XOCP message is assembled, the field for the trading partner recipient can be left blank, or populated with a name or an XPath expression. For example, with the extended functionality provided by the B2B integration plug-in, the WebLogic Integration Studio allows you to specify either a trading partner name or an XPath expression when you configure the routing expression for a send business message action. (For details, see [“Sending and Receiving Business Messages”](#) in *Creating Workflows for B2B Integration*.)
- *XPath expressions associated with a trading partner*
Trading partner XPath router expressions and trading partner XPath filter expressions can be associated with a trading partner through the B2B Console.
- *XPath expressions associated with a business protocol definition*
Business protocol XPath router expressions and business protocol XPath filter expressions can be associated with the XOCP business protocol definition through the B2B Console.

Note: The use of XPath expressions for routing messages to trading partners is a special feature of the XOCP business protocol. When an application sends a message using RosettaNet or cXML, the target recipient is explicitly encoded.

As described in “XOCP Hub and Spoke Delivery Channels” on page 1-7, XOCP delivery channels are configured as hub delivery channels or spoke delivery channels. When a business message is transmitted to a hub delivery channel, the XOCP router logic plug-in generates an XML message-context document. The message-context document captures properties associated with the trading partner sender and recipients, as identified by any conversation definition and collaboration protocol agreements. Any defined XPath router expressions use the XPath syntax to select a subset of trading partners from the set of trading partners and associated properties captured in the message-context document. The selected trading partners are the intended recipients of the XOCP business message.

The following figure provides a high-level overview of message processing and routing on an XOCP hub delivery channel.

Figure 3-1 Message Processing in XOCP Hub Delivery Channel



As noted, when an incoming message is received by a hub delivery channel, the conversation definition and any associated collaboration agreements that have been configured are used to identify the recipient trading partners to be included in the message-context document.

For example, suppose you have a Query Price and Availability conversation definition (QPA) that defines two roles: buyer and supplier. For trading partner 1 (TP1), a hub delivery channel, `tp1-hub-dc`, is defined. For each of the other four trading partners

(TP2, TP3, TP4, and TP5) a spoke delivery channel is defined: `tp2-spoke-dc`, `tp3-spoke-dc`, `tp4-spoke-dc`, and `tp5-spoke-dc`. The following collaboration agreements are defined on TP1:

- TP1 (`tp1-hub-dc`) is assigned to the QPA role of supplier.
TP2 (`tp2-spoke-dc`) is assigned to the QPA role of buyer.
- TP1 (`tp1-hub-dc`) is assigned to the QPA role of buyer.
TP3 (`tp3-spoke-dc`) is assigned to the QPA role of supplier.
TP4 (`tp4-spoke-dc`) is assigned to the QPA role of supplier.
TP5 (`tp5-spoke-dc`) is assigned to the QPA role of supplier.

When a business message from TP2 is sent to the supplier role, and it is received on `tp1-hub-dc`, the collaboration agreement in which `tp1-hub-dc` is assigned the buyer role is identified. That collaboration agreement is then used to identify the trading partner recipients for the business message.

In this case, the XML message-context document generated by the XOCF router logic plug-in contains identifying information and properties for both the sender (TP1) and the recipients (TP3, TP4, and TP5).

Any XPath router expressions that are defined are used by the XOCF router logic plug-in to select trading partners from this list. The trading partners selected by the XPath expression(s) are then included in the message routing header. Each XPath expression is configured to replace or to be appended to the results of the previous expression. For additional information about how XPath router expressions are processed, see “XPath Router Expression Processing” on page 3-6.

Note: Although the `context` attribute of the `wlc` element in the message-context document is updated, in the course of processing, by the XPath router logic plug-in (from `message-router` to `trading-partner-router`, then to `hub-router`), no other element of the document is changed. In other words, all XPath router expressions are used to select from the set of trading partners that was originally included.

Once the XOCF router logic plug-in finishes processing and messages are routed to the specified trading partners, the XPath filter logic plug-in generates an XML message-context document for each outgoing business message.

If XPath filter expressions are defined, they are evaluated against the message-context document generated by the XOCF filter to determine whether the business message should be sent to a trading partner or not. XPath filter expressions must evaluate to a boolean true or false.

- If *any expression* evaluates to *false*, then the message is not sent and no other expressions are evaluated.
- If *all expressions* evaluate to *true*, the message is processed normally.

The order in which these XPath filter expressions are evaluated is described in “XPath Filter Expression Processing” on page 3-7.

As noted in “Overview of Advanced Features” on page 3-2, XPath router and filter expressions can reference user-defined extended properties. (For a discussion of extended properties for trading partners, see “Trading Partner Extended Properties” on page 3-13.)

XPath Router Expression Processing

XPath router expressions fall into three categories. Each expression is configured to replace or to be appended to the results of the previous XPath expression. The expressions are processed in the following order:

1. *Message XPath router expressions*
Message XPath router expressions are included in the business message. They always apply to the routing of that business message.
2. *Trading Partner XPath router expressions*
Trading partner XPath router expressions are associated with the sending trading partner. They apply to all messages sent by that trading partner.
3. *Business Protocol XPath router expressions*
Business protocol XPath router expressions apply to all incoming business messages using a particular protocol.

If you define more than one trading partner XPath router expression or business protocol XPath router expression, all the trading partner XPath router expressions are evaluated before the business protocol XPath router expressions. Expressions of the same type are processed in the order listed in the B2B Console.

XPath Filter Expression Processing

WebLogic Integration supports XPath filter expressions for two entities: trading partners and business protocols. Expressions are evaluated in the following order:

1. *Trading Partner XPath filter expressions*
These expressions are associated with the recipient trading partner. They apply to all messages sent to that trading partner.
2. *Business Protocol XPath filter expressions*
These expressions apply to all outgoing business messages using the specified protocol.

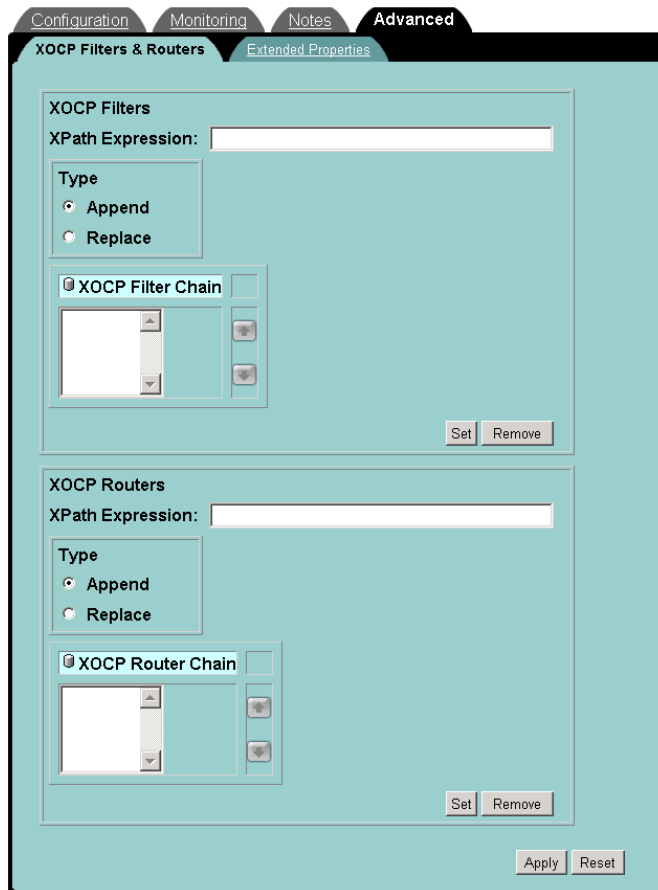
If you define more than one XPath filter expression for a trading partner or a business protocol, all the XPath filter expressions for trading partners are evaluated before those for business protocols. Processing continues until an expression evaluates to false, or until all expressions are processed. Expressions of the same type are processed in the order listed in the B2B Console.

Configuring Router and Filter Expressions

You can configure XPath router and filter expressions for both trading partners and business protocols from the B2B Console. To configure XPath router and filter expressions for a trading partner, complete the following steps:

1. In the B2B Console, open the Trading Partner page and select the Advanced tab. Two nested tabs, XOCP Filters & Routers and Extended Properties, are displayed.
2. Select the nested XOCP Filters & Routers tab. It is displayed as shown in the following figure.

Figure 3-2 XOCP Filters & Routers Tab



From the XOCP Filters & Routers tab you can configure and order required XPath expressions. The detailed information required to perform these tasks is provided by online help. See “Getting Help” on page 2-7.

A Filters & Routers tab that is identical to the preceding tab is available when you select the Configuration tab on the XOCP business protocol definition page. Only the context differs for the two pages (the tab available from the XOCP business protocol definition page is used to define filters and routers that apply to all messages, not just those that apply to a particular trading partner).

Additional Information

For more information about routing and filtering business messages, the structure of message-context documents, and creating XPath expressions, see “Routing and Filtering Business Messages” in *Programming Logic Plug-Ins for B2B Integration*.

Custom Logic Plug-Ins

Logic plug-ins are Java classes that can intercept and process business messages at run time. Each business protocol is associated with three standard logic plug-ins:

- *Router logic plug-in*
This logic plug-in processes incoming business messages and generates the message-context document that is used in the processing. By default, this logic plug-in is the first in the router chain. The XOCP router logic plug-in for a hub delivery channel can modify the list of trading partner recipients based on XPath router expressions, as described in “XPath Expressions in Routing and Filtering” on page 3-3.
- *Router enqueue logic plug-in*
This logic plug-in adds business messages to the router message queue. By default, this logic plug-in is the last in the router chain.
- *Filter logic plug-in*
This logic plug-in processes outgoing business messages and generates the message-context document that is used in processing outgoing messages. By default, this logic plug-in is the only one in the filter chain. The XOCP filter logic plug-in for a hub delivery channel can use any defined XPath filter expressions to determine whether to send a message, as described in “XPath Expressions in Routing and Filtering” on page 3-3.

The following table describes built-in logic plug-ins.

Table 3-1 Business Protocol Logic Plug-Ins

Protocol	Processing Chain	Logic Plug-In
XOCP	Router	XOCP router
		XOCP router enqueue
	Filter	XOCP filter
RosettaNet	Router	RosettaNet router
		RosettaNet router enqueue
	Filter	RosettaNet filter
cXML	Router	cXML router
		cXML router enqueue
	Filter	cXML filter

Custom logic plug-ins can be developed and added to either a router or a filter processing chain for a business protocol. Inclusion in such a processing chain, however, does not necessarily limit the functionality of a logic plug-in. Although custom logic plug-ins are associated with one of these two types of processing chain, they are not required to perform routing or filtering services. Thus, for example, a custom logic plug-in might be developed to examine message content and capture information for billing purposes.

Configuring Logic Plug-Ins

After you develop a custom logic plug-in, you must add a definition for it to a business protocol definition router or filter chain. To do so, open the WebLogic Integration B2B Console and complete the following procedure:

1. Create a definition for the logic plug-in, specifying the following properties:
 - Name of the logic plug-in
 - Plug-in type (router chain or filter chain)
 - Java class that implements the logic plug-in interface
 - Parameter name/value pairs to be used for initializing the Java class

To create a logic plug-in definition:

- a. Select Logic Plug-Ins from the navigation tree. The Logic Plug-Ins page is displayed in the right pane.
 - b. Select Create a New Logic Plug-In. The logic plug-in page is displayed. It includes fields in which you can specify the required properties.
2. Add the logic plug-in definition to the business protocol definition and specify the position of the logic plug-in in the router or filter chain.

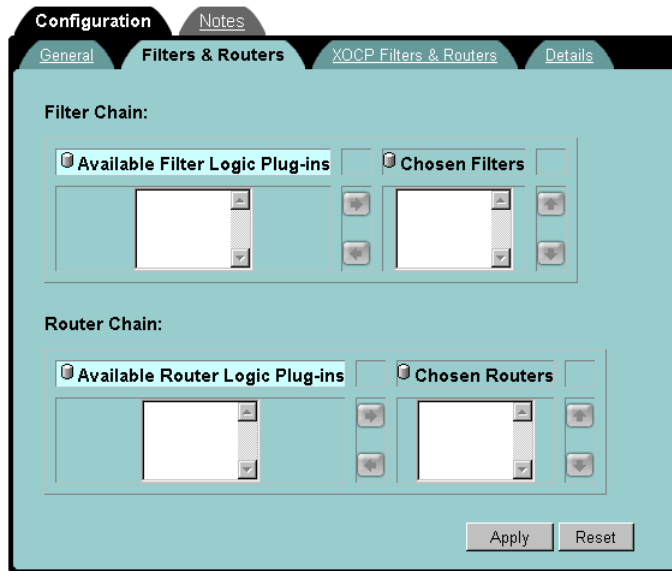
Step 2 is performed from the business protocol page, as described in the following section.

Adding a Custom Logic Plug-In to a Business Protocol Router or Filter Chain

To add a logic plug-in to a business protocol router or filter processing chain:

1. Select a business protocol from the navigation tree. Two high-level tabs, Configuration and Notes, are displayed in the right pane.
2. From the Configuration tab, select the nested Filters & Routers tab. It is displayed as shown in the following figure.

Figure 3-3 Filters & Routers Tab for Business Protocols



From the Filters & Routers tab you can select and order any required logic plug-ins that are available. The detailed information required to perform these tasks is provided by online help. See “Getting Help” on page 2-7.

Additional Information

For more information about the logic plug-in API, and for guidelines for developing and deploying custom logic plug-ins, see [Programming Logic Plug-Ins for B2B Integration](#).

Trading Partner Extended Properties

The default properties associated with a trading partner can be augmented to support application-specific requirements through the use of trading partner extended properties. You can add extended properties from the B2B Console. Once added, these properties are included in any message-context documents generated by the business protocol router and filter logic plug-ins as uniquely named extended property sets.

Extended property sets are modeled in the repository so they can be retrieved as subtrees within an XML document. These XML subtrees appear in the message-context XML document generated by the built-in router and filter logic-plug-ins. XPath expressions can reference these extended properties. The root elements of each extended property set associated with a given trading partner are inserted as the last children of the <trading-partner> element node. The following example shows an XML document generated from the repository with an extended property set:

```
<wlc context="message-router">
...
<trading-partner name="ABC International"
email="admin@abc.com"
phone="+1 123 456 7890">
<address>123 ABC Street., Anytown, CA 95131</address>
<extended-property-set name="ABC Contact">
    <business-contact>Joe Smith</business-contact>
    <phone type="work">+1 123 456 7654</phone>
    <phone type="cell">+1 321 654 4567</phone>
    <city>Anytown</city>
    <state>California</state>
</extended-property-set>
</trading-partner>
...
</wlc>
```

Configuring Trading Partner Extended Properties

To add extended properties to a trading partner:

1. Select a trading partner from the navigation tree. Four high-level tabs (Configuration, Monitoring, Notes, and Advanced) are displayed in the right pane.
2. Select the Advanced tab. Two tabs (XOCP Filters & Routers and Extended Properties) are nested on the Advanced tab.
3. Select the Extended Properties tab. It is displayed as shown in the following figure.

Figure 3-4 Trading Partner Extended Properties Tab

The screenshot shows a web application interface with a teal background. At the top, there are four main tabs: Configuration, Monitoring, Notes, and Advanced. The Advanced tab is selected, and it contains two sub-tabs: XOCP Filters & Routers and Extended Properties. The Extended Properties tab is active, displaying the following fields and controls:

- Property Name:
- Property Value:
- Attributes section:
 - Name:
 - Value:
 - A list box labeled "Attributes" with a plus icon and a minus icon.
 - Buttons: Set, Remove
- Extended Properties section:
 - A list box labeled "Extended Properties" with a plus icon and a minus icon.
 - Buttons: Add/Apply, Remove, Reset

Using this tab you can set the required extended properties. The detailed information required to perform these tasks is provided by online help. See “Getting Help” on page 2-7.

Additional Information

For more information about the structure of message-context documents and creating XPath expressions to reference extended properties, see [*Programming Logic Plug-Ins for B2B Integration*](#).

4 Importing and Exporting B2B Integration Components

Before trading partners can participate in conversations hosted by a WebLogic Integration B2B application, they must set up their environments to meet the requirements of the application. Specifically, they must populate their systems with various components on which a B2B application depends, such as trading partner definitions, conversation definitions, collaboration agreements, and workflows. You can facilitate this aspect of environment setup by creating the necessary B2B components and delivering them to your trading partners.

This section explains how to export and import the components necessary to implement B2B applications. It includes the following topics:

- B2B Integration Components
- Export and Import Overview
- Exporting from the B2B Console
- Importing to the B2B Console
- Exporting a Workflow Package
- Importing a Workflow Package

B2B Integration Components

The components of a WebLogic Integration B2B application include:

- Trading partner definitions
- Collaboration agreements
- Conversation definitions
- Collaborative workflows

A collaborative workflow includes the template definition and any associated:

- Business operations
- Business calendars
- Plug-ins
- Event key tables
- XML entities (for example, XML documents, schema files, or XSLT templates)

It is common for multiple trading partners to use the same component. For example, all parties who are assigned the same role in a collaboration agreement for a given conversation definition require the same conversation definition and public workflow template. To participate in a peer-to-peer exchange, each trading partner must have a trading partner definition for itself and the other party.

To facilitate trading partner setup, one partner can define the required components, and then export them so they become available for import by other trading partners. Trading partner definitions, collaboration agreements, and conversation definitions can be exported and imported through the WebLogic Integration B2B Console. Collaborative workflows can be exported and imported through the WebLogic Integration Studio.

Although many components of a business collaboration are identical when used by different trading partners, some require minor modification after import. For example, if a trading partner definition for a remote trading partner is exported by a local trading partner and then imported by the remote trading partner, the remote trading partner must change the value of the trading partner type property from remote to local before the imported definition can be used.

Workflow templates and conversations typically require modification of the specified organization because different trading partners have different organizational structures in the Studio.

Export and Import Overview

To export the components required by a participant in a B2B exchange:

1. Export the required trading partner definitions, conversation definitions, and collaboration agreements from the B2B Console.

Typically, you can simply export the required collaboration agreement with the Export all referenced entities option set. This option ensures that all necessary conversation definitions and trading partners are exported along with the specified collaboration agreement.

Note: When you use the Export all referenced entities option, only the referenced objects are exported. For example, although many delivery channels may be associated with a particular trading partner definition, when you export a collaboration agreement with the Export all referenced entities option set, only those parts of the trading partner definition required for the collaboration agreement (that is, only the required party identifiers, delivery channels, document exchanges, transports, and certificates) are exported.

The export procedure is described in “Exporting from the B2B Console” on page 4-4.

2. Export the collaborative workflow package required by the trading partner to implement its role in the conversation.

The workflow export procedure is described in “Exporting a Workflow Package” on page 4-9.

To import and modify the exported components for use:

1. Import the required trading partner definitions, conversation definitions, and collaboration agreements.

One or more XML files may be provided. The import procedure is described in “Importing to the B2B Console” on page 4-8.

2. Verify the imported components by viewing them in the B2B Console. Make adjustments as required for use. For example:
 - Reset the trading partner type property for each imported trading partner (from local to remote, or vice versa). Typically, the trading partner type in an imported trading partner definition does not reflect the type required on your system.
 - Update the certificates and security configuration as required.
 - Reset the organization specified for the workflow template in the conversation definition. Typically, the organizations defined in the Studio on your system are not the same as those defined for other trading partners. The organization specified should be the organization to which the workflow is assigned.
3. Import the collaborative workflow package required to implement your role in the conversation.

The workflow import procedure is described in “Importing a Workflow Package” on page 4-12.

Exporting from the B2B Console

To export entities from the B2B Console:

1. Click the B2B node in the navigation tree.

The B2B page is displayed in the right pane. It contains several high-level tabs.

2. Select the high-level Configuration tab.
3. Select the nested Export tab.

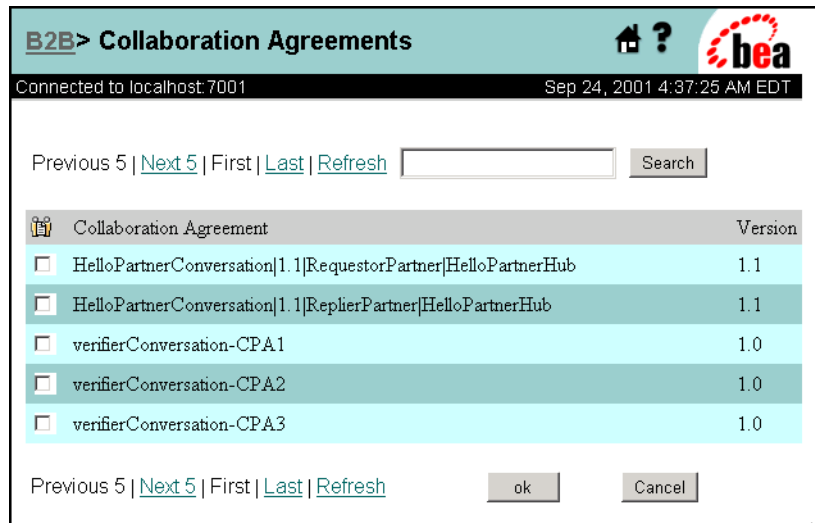
The export options are displayed, as shown in the following figure.

Figure 4-1 B2B Export Tab

The screenshot displays the B2B Export configuration interface. At the top, there are tabs for Configuration, Monitoring, and Notes. Under Configuration, there are sub-tabs for General, Security, Proxy, Import, Export, and Preferences. The Export sub-tab is selected. The main content area is divided into sections: 'Scope of Export' with checkboxes for All, B2B, Trading Partners, Conversation Definitions, Collaboration Agreements, Business Protocol Definitions, and Logic Plug-Ins, each with a corresponding 'Browse...' button; 'Format' with radio buttons for Standard (selected) and Extensive; an 'Encoding:' text input field; and a checked checkbox for 'Export all referenced entities'. At the bottom right, there are 'Export' and 'Reset' buttons.

4. Select the entities to be exported by performing one or more of the following steps:
 - To export all repository data, select the All option.
 - To export all instances of a particular entity (such as Trading Partners), select the name of the appropriate entity.
 - To select individual instances of a particular entity, display the list of available instances by clicking the Browse button to the right of the appropriate entity name. A page showing a selection of the available instances of the specified entity is displayed, as shown in the following figure.

Figure 4-2 Entity Selection Page



Select individual instances of the entity by selecting them on this page. Then click the OK button to return to the Export tab.

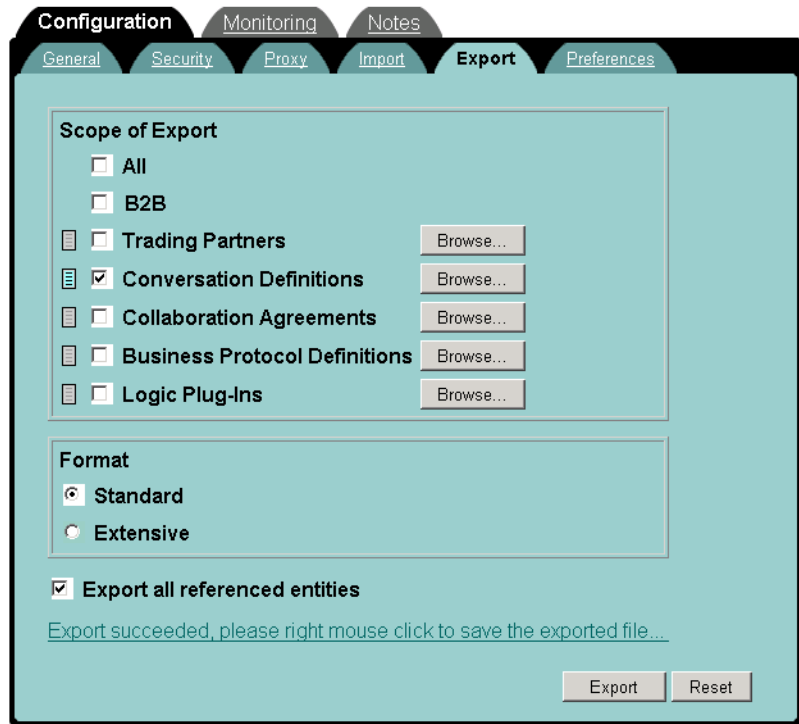
5. On the Export tab, select a format.

The default format is standard. The extensive format includes system information, such as the total number of updates (update-count) and timestamps.

6. If you want all entities referenced by the selected entities to be included, select the Export all referenced entities option.
7. Click the Export button.

The specified entities are exported to an XML file. When the export is complete, a message is displayed as a link at the bottom of the page, as shown in the following figure.

Figure 4-3 Notification of Successful Export



8. Right-click the link. A menu of shortcuts is displayed. Select Save Link As from the shortcut menu.

The Save As dialog box is displayed, allowing you to select the location or rename the file. (The default filename is `exportConfig.xml`.)

9. When you have selected a target directory, click Save.

Importing to the B2B Console

Before importing data, you must shut down the B2B engine, as described in [“Starting and Stopping the B2B Engine from the B2B Console”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

To import to the B2B Console:

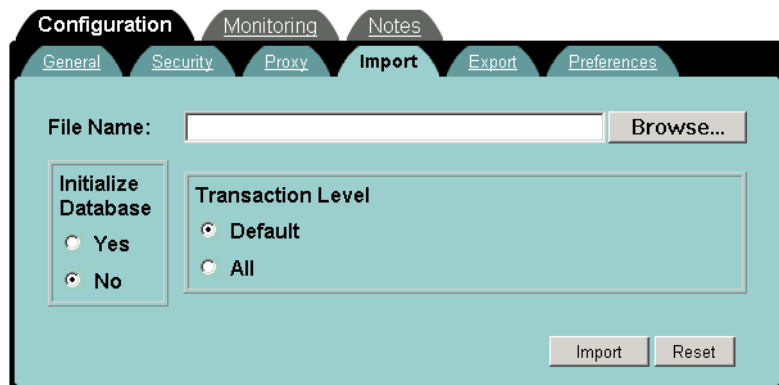
1. Click the B2B node in the navigation tree.

The B2B page is displayed in the right pane. It contains several high-level tabs.

2. Select the high-level Configuration tab.
3. Select the nested General tab.
4. Verify that the status is inactive.
5. Select the high-level Configuration tab. Then select the nested Import tab.

The import options are displayed, as shown in the following figure.

Figure 4-4 B2B Import Tab



The screenshot shows the B2B Import Tab interface. At the top, there are tabs for Configuration, Monitoring, and Notes. Under Configuration, there are sub-tabs for General, Security, Proxy, Import, Export, and Preferences. The Import tab is selected. The interface includes a 'File Name' field with a 'Browse...' button. Below this are two sections: 'Initialize Database' with radio buttons for 'Yes' and 'No' (where 'No' is selected), and 'Transaction Level' with radio buttons for 'Default' and 'All' (where 'Default' is selected). At the bottom right are 'Import' and 'Reset' buttons.

6. Select the Browse option beside the File Name field.

The Upload File dialog box is displayed.

7. Select the XML file that contains the data to be imported, then click the Open button.

8. Verify that Initialize Database is set to No (unless you want to delete all existing data).
9. Select the Transaction level:
 - *Default*—A transaction is initiated for each entity. All instances of the same entity are imported in a single transaction. (For example, if the entity in question is Trading Partner, then all trading partners are imported.) If invalid data is detected during any single transaction, that transaction is rolled back and processing continues for the next entity type.
 - *All*—The data in the selected file is imported as a single transaction. If invalid data is detected, the entire transaction is rolled back.
10. To import the data, click Import.

Note: If an entity has the same name as an entity in the repository, it is overwritten.
11. Restart the B2B engine, as described in [“Restarting the B2B Engine from the B2B Console”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Exporting a Workflow Package

The WebLogic Integration Studio allows you to select and export template definitions and related workflow components. The components you select are packaged in a Java Archive (JAR) file known as a workflow package.

When you export a workflow package, you can:

- Use passwords to prevent unauthorized access to the contents of the package.
- Publish the package to make the components available only for reading. When you publish a package, trading partners can employ the workflow components, but they are unable to modify them.

The Studio supports the export of template definitions, business operations, business calendars, plug-ins, event key tables, and XML repository items. The only elements that cannot be exported are the organizations, users, and roles defined in the Studio. When you export a template definition, it is disassociated from the current organization. When a template definition is imported into a target system, a new organization must be selected for the template.

To export a workflow package from the Studio:

1. Start the Studio as described in [“Starting the Studio”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

2. Choose Tools→Export Package from the Menu bar.

The Export: Select File dialog box is displayed.

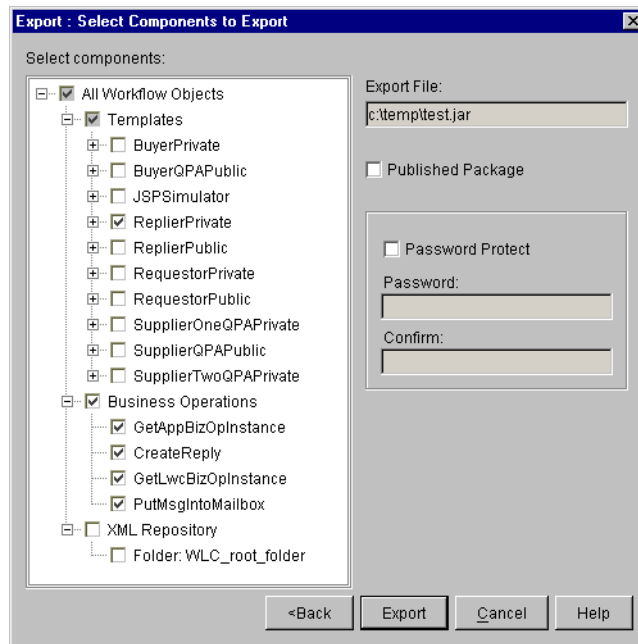
3. Enter the location and filename of the JAR file to be created.

Note: If you specify a JAR file by selecting it from a list invoked through the Browse option, the export overwrites the file.

4. Click Next.

The Export: Select Components to Export dialog box is displayed, as shown in the following figure.

Figure 4-5 Export: Select Components to Export Dialog Box



The dialog box displays a list of all items currently defined. When you select an item, items referenced by it are automatically selected.

Note: Items in the XML repository are not automatically selected; you must select them manually.

5. Select the items to export.
6. Click Export.

When the export operation is complete, the Export: Review Export Summary dialog box displays the following message:

Components were exported to JAR file: *path/filename.jar*

7. Click Close to dismiss the dialog box.

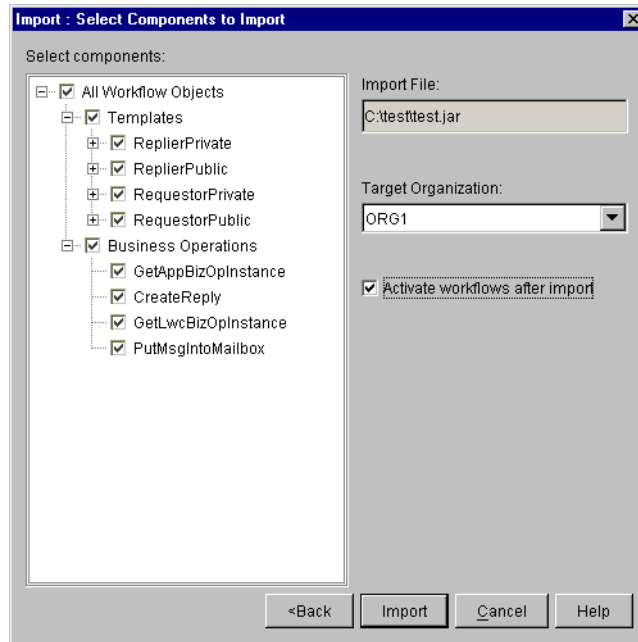
Importing a Workflow Package

To import a workflow package from the Studio:

1. Start the Studio as described in “[Starting the Studio](#)” in *Starting, Stopping, and Customizing BEA WebLogic Integration*.
2. Choose Tools→Import Package from the Menu bar.
The Import: Select File dialog box is displayed.
3. Click Browse.
The Open dialog box is displayed.
4. Select the file to import, and then Click Open.
You are returned to the Import: Select File dialog box.
5. Click Next.

The Import: Select Components to Import dialog box is displayed, as shown in the following figure.

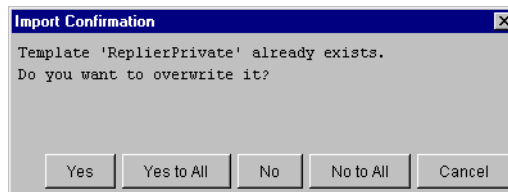
Figure 4-6 Import: Select Components to Import Dialog Box



The dialog box lists all the items contained in the JAR file.

6. Deselect any items you do not want to import.
7. Expand the drop-down list and select a target organization.
8. To avoid having to activate the workflows after the import, select the Activate workflows after import option.
9. Click Import.

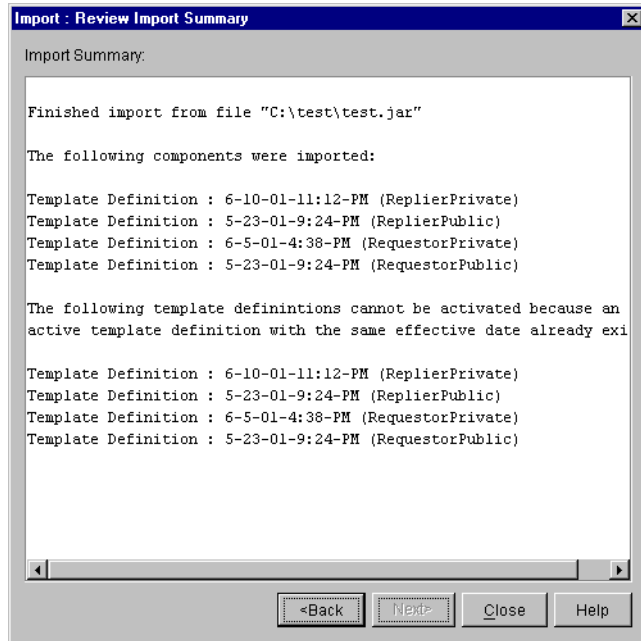
If the name of any item is the same as that of an existing item, the Import Confirmation dialog box is displayed, as shown in the following figure.



4 Importing and Exporting B2B Integration Components

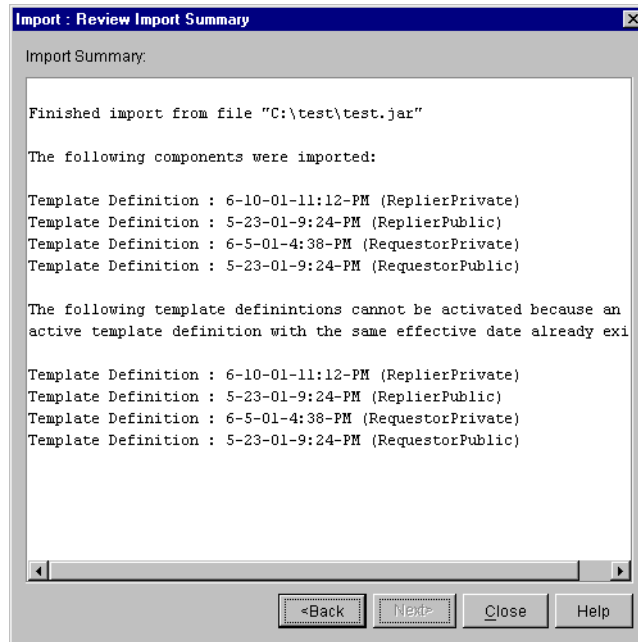
10. When complete, the Import: Review Import Summary dialog box is displayed, as shown in the following figure.

Figure 4-7 Import: Review Import Summary Dialog Box



11. Click Close to dismiss the dialog box.

Figure 4-8 Import: Review Import Summary



12. Click Close to dismiss the dialog box.

4 *Importing and Exporting B2B Integration Components*

5 Monitoring B2B Integration

This section provides an overview of how you can use the B2B Console to monitor and control WebLogic Integration, trading partner sessions, delivery channels, conversations, and collaboration agreements. It includes the following topics:

- Overview of Monitoring
- Note About Conversation Monitoring
- B2B Console Monitoring Pages
- Monitoring the B2B Engine
- Monitoring Trading Partner Sessions
- Monitoring Delivery Channels
- Monitoring Conversations
- Monitoring Collaboration Agreements
- Monitoring Messages

Overview of Monitoring

The WebLogic Integration B2B Console enables you to control and monitor the following entities:

- *B2B engine*—WebLogic Integration process that provides and controls functionality for B2B integration.
- *Trading partner session*—Connections between trading partner delivery channels. One or more conversations may be associated with a single connection.
- *Delivery channels*—Connection endpoints in a trading partner session.
- *Conversations*—Exchange of messages associated with trading partner delivery channels and roles defined in a collaboration agreement.
- *Collaboration agreements*—Contracts in which trading partners specify the parameters of their message exchanges.

Note About Conversation Monitoring

The primary benefit of being able to monitor conversations from the B2B Console is that it makes it possible for a conversation initiator to terminate a conversation and for other participants to exit the conversation.

cXML and RosettaNet conversations, which are not monitored via the B2B Console, do not support this conversation termination feature. The ability to view a list of conversations and select one to be either terminated (from the B2B engine running on the initiating trading partner system) or exited (from the B2B engine running on the participant system) is limited to the XOCP protocol.

In RosettaNet and cXML, control resides at the level of the delivery channel and the trading partner session. Throughout this section, any reference to the display of conversations applies to XOCP delivery channels and trading partner sessions only.

B2B Console Monitoring Pages

Monitoring functions can be invoked from the Monitoring tab, which, in turn, is available from the pages of the WebLogic Integration B2B Console listed in the following table.

Table 5-1 Accessing the B2B Console Pages

This page . . .	Is displayed in the B2B Console when you select . . .
B2B page	B2B from the navigation tree
Trading partner page	A trading partner from either the navigation tree or the Trading Partners page.
Conversation page	A conversation definition from either the navigation tree or the Conversations page.
Collaboration agreement page	A collaboration agreement from either the navigation tree or the Collaborations Agreements page.

For a detailed description of the B2B Console, including its support for monitoring tasks, see “Overview of WebLogic Integration B2B Console” on page 2-2.

The following table summarizes the functions available on each page.

Table 5-2 Monitoring WebLogic Integration

Select the Monitoring tab on the . . .	Then select this nested tab . . .	To access these functions . . .
B2B page	General	<ul style="list-style-type: none"> ■ Status (running or inactive). ■ Shutdown or Restart (depending on the current status of the B2B engine).
	Statistics	<ul style="list-style-type: none"> ■ Summary statistics (number of trading partner sessions, active collaboration agreements, active conversations, active delivery channels, and messages sent and received, and time of the last message sent and received).
	Log	<ul style="list-style-type: none"> ■ View the B2B engine log. ■ Set view options for the log (level of logging, number of lines per page).
Trading partner page	Sessions	<ul style="list-style-type: none"> ■ View a list of run-time sessions for the selected trading partner (session identifier and start time are listed). ■ View the details of a selected session (status, start time, number of conversations, number of messages sent, number of messages outstanding, times at which messages were last sent and received, times of first and last failed message). ■ From the details about a selected session, link to a list of either the constituent conversations or the outstanding messages.
	Delivery Channels	<ul style="list-style-type: none"> ■ View a list of the delivery channels associated with the selected trading partner. ■ View details about a selected delivery channel (status, trading partner sessions, conversations, collaboration agreements, messages sent). ■ From the details about a selected delivery channel, modify the status (enable or disable) or link to a list of trading partner sessions, collaboration agreements, conversations, or messages sent. For each list, additional detail is available.

Table 5-2 Monitoring WebLogic Integration (Continued)

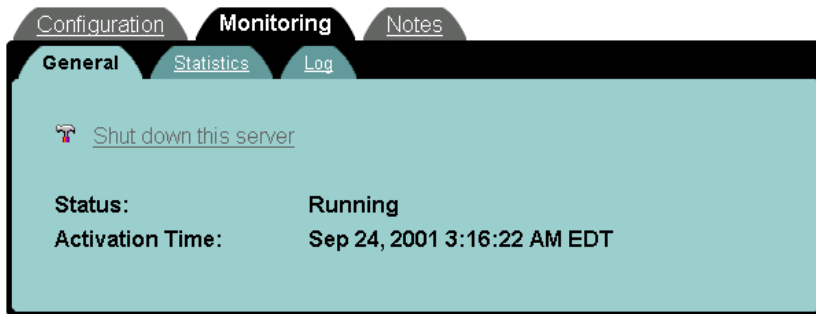
Select the Monitoring tab on the . . .	Then select this nested tab . . .	To access these functions . . .
Conversation page	General	<ul style="list-style-type: none"> ■ View a list of the active conversations for the selected conversation definition. (A conversation identifier and the start time are listed.) ■ View details of a selected conversation (start time, self-initiated indicator, time of last message, identity of last sender). ■ End/Leave a conversation. <p>These features are available only for XOCP. See “Note About Conversation Monitoring” on page 5-3.</p>
Collaboration agreement page	General	<ul style="list-style-type: none"> ■ View identifying information (name, version, business protocol, associated conversation definition, and number of parties) ■ View the status (enabled/disabled, registered/unregistered) ■ Modify the status (enable/disable, register/unregister)

The following sections provide an overview of each type of entity you can monitor from the WebLogic Integration B2B Console. The detailed information required to navigate the B2B Console to view available monitoring options is provided in the online help. See “Getting Help” on page 2-7.

Monitoring the B2B Engine

When you select B2B from the navigation tree, a set of three high-level tabs (Configuration, Monitoring, and Notes) is displayed in the right pane. If you select the Monitoring tab, and then select the nested General tab, the General tab is displayed as shown in the following figure.

Figure 5-1 General Tab for Monitoring the Server



Status information and the time at which the server started are displayed.

To shut down the server, select `Shut down this server`. The *Terminate* shutdown option is displayed. This option shuts down active delivery channels, triggering the termination of the associated trading partner sessions. The termination of the trading partner sessions triggers, in turn, the termination of associated conversations and the removal of queues.

Note: For information about shutting down the B2B engine, see [“Starting and Stopping the B2B Engine from the B2B Console”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

To display summary statistics for your server, select the Statistics tab, as shown in the following figure.

Figure 5-2 Server Statistics



Monitoring Trading Partner Sessions

A trading partner session is a connection between trading partner delivery channels. WebLogic Integration supports two options for listing active trading partner sessions:

- *List the trading partner sessions for a selected trading partner*
To do so, select a trading partner from the navigation tree or Trading Partners page. Then, in the right pane, select the Monitoring tab. Finally, select the Sessions tab. A list of active trading partner sessions for the specified trading partner is displayed.
- *List the trading partner sessions for a selected delivery channel*
To do so, select a trading partner from the navigation tree or Trading Partners page. Then, in the right pane, select the Monitoring tab. Finally, select the Delivery Channels tab. A list of the active delivery channels for the specified trading partner is displayed.

As described in “Monitoring Delivery Channels” on page 5-11, the number of trading partner sessions for the specified delivery channel is displayed as one of the summary statistics for the delivery channel. You can view a list of the trading partner sessions for the delivery channel by clicking that number.

When you select a trading partner session from a list of sessions, the following information is displayed.

Figure 5-3 Monitoring a Trading Partner Session

The screenshot displays a web interface for monitoring a trading partner session. The breadcrumb navigation at the top reads: **B2B > Trading Partners > Trading Partner > Trading Partner Session**. On the right side of the header, there is a home icon, a question mark, and the BEA logo. Below the header, a status bar indicates the connection: **Connected to localhost: 7001** on the left and **Sep 24, 2001 5:12:22 AM EDT** on the right. A link with a trash icon is labeled **Shut Down this Trading Partner Session**. The main content area shows the following session details:

Trading Partner Session:	HelloPartnerHub
Status:	Active
Start Time:	
Conversations :	0
Messages Sent:	3
Messages Outstanding:	0
Last Message Sent:	Sep 24, 2001 5:11:53 AM EDT
Last Message Received:	Sep 24, 2001 5:11:50 AM EDT
First Failed Message:	
Last Failed Message:	

Summary statistics for the trading partner session are displayed. From the summary, you can link to either a list of the active conversations for the session or a list of the outstanding messages for it.

Note: For successful deployment, the delivery channel endpoints for the session must be bound to the same business protocol. (Protocol binding is assigned in the document exchange assigned to the delivery channel.)

To shut down a trading partner session, select *Shut Down this Trading Partner session*. Shutdown of the trading partner session triggers termination of the associated conversations and removal of queues.

Monitoring Delivery Channels

A delivery channel is a connection endpoint in a trading partner session. To display a list of the active delivery channels for a trading partner, complete the following procedure:

1. Select a trading partner from the Trading Partners page or the navigation tree.
2. Select the Monitoring tab, and then select the Delivery Channels tab.

When you select a delivery channel from the list, the following types of information are displayed.

Figure 5-4 Monitoring a Delivery Channel

The screenshot shows a web interface for monitoring a delivery channel. At the top, there is a breadcrumb trail: **B2B > Trading Partners > HelloPartnerHub > HelloPartnerHubDeliveryChannel**. To the right of the breadcrumb are icons for home, help, and the BEA logo. Below the breadcrumb is a status bar: **Connected to localhost:7001** on the left and **Sep 24, 2001 5:16:47 AM EDT** on the right. The main content area features a link with a speaker icon: [Disable this Delivery Channel](#). Below this is a summary table:

Delivery Channel:	HelloPartnerHubDeliveryChannel
Status:	Enabled
Trading Partner Sessions:	2
Conversations :	0
Collaboration Agreements:	2
Messages Sent:	0

Status information and summary statistics for the designated delivery channel are displayed. From the summary, you can link to lists of trading partner sessions, collaboration agreements, conversations, and messages sent.

You can disable a delivery channel by selecting [Disable this Delivery Channel](#). Disabling a delivery channel triggers termination of all active trading partner sessions associated with the delivery channel. Termination of the trading partner sessions, in turn, triggers termination of the associated conversations and removal of queues.

Monitoring Conversations

Note: As discussed in “Note About Conversation Monitoring” on page 5-3, support for monitoring conversations is available only for XOCP trading partner sessions.

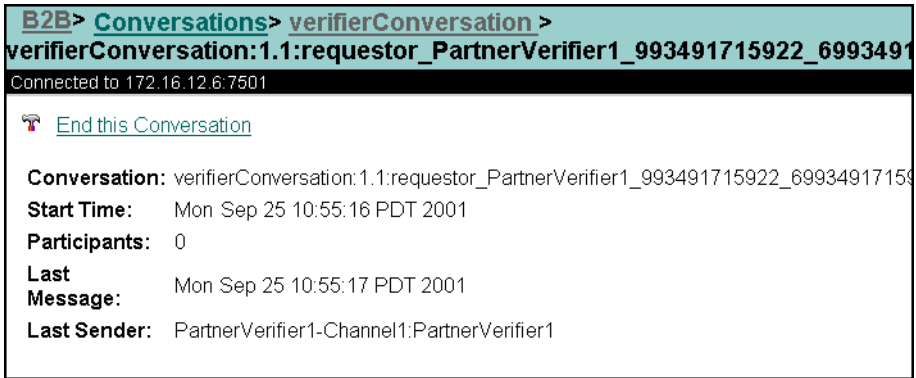
WebLogic Integration allows you to display a list of the active XOCP conversations for three entities: conversations, delivery channels, and trading partner sessions. Follow the appropriate instructions in the following table.

Table 5-3 Listing Conversations

To list conversations for a particular . . .	Perform the following steps . . .
Conversation definition	<ol style="list-style-type: none"> 1. Select a conversation definition from the navigation tree or Conversations page. 2. Select the Monitoring tab. A list of active conversations for the designated conversation definition is displayed.
Delivery channel	<ol style="list-style-type: none"> 1. List the delivery channels as described in “Monitoring Delivery Channels” on page 5-11. When you select a delivery channel from the list, the number of conversations for that delivery channel is displayed as one of the summary statistics for the delivery channel. 2. To view a list of the conversations for the specified delivery channel, click the number representing the number of conversations in the summary statistics.
Trading partner session	<ol style="list-style-type: none"> 1. List trading partner sessions as described in “Monitoring Trading Partner Sessions” on page 5-9. When you select a trading partner session from the list, the number of conversations for that session is displayed as one of the summary statistics for the trading partner session. 2. To view a list of the conversations for the specified delivery channel, click the number representing the number of conversations in the summary statistics.

When you select a conversation from a list, the following is displayed.

Figure 5-5 Monitoring a Conversation



The following information is displayed: identifying information, the starting time, a self-initiated indicator, the time of the last message, and the identity of the last sender.

If the local trading partner initiates the conversation, that trading partner can end the conversation by selecting `End this Conversation`.

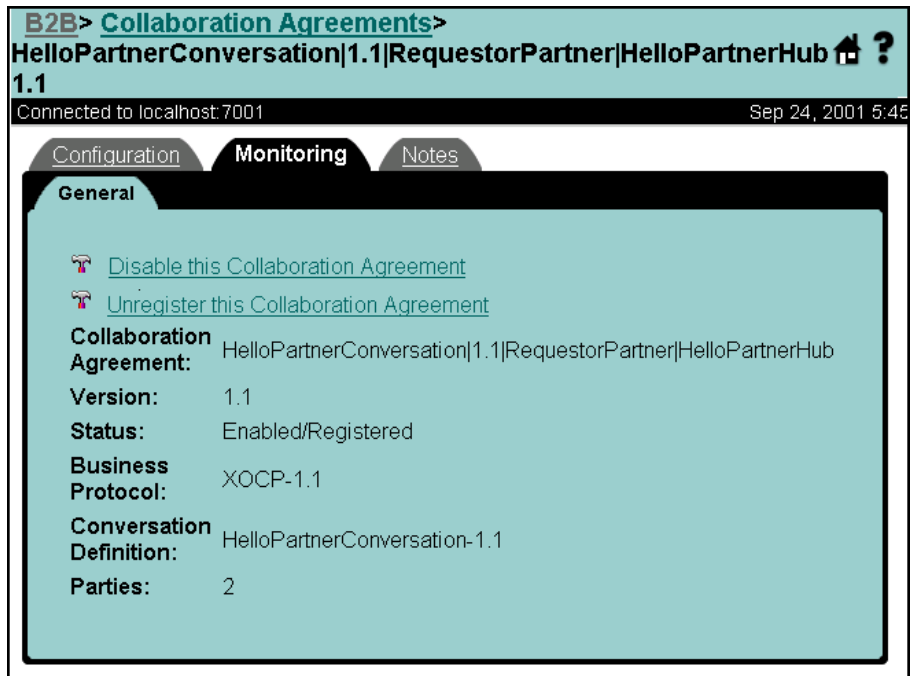
If the local trading partner does not initiate the conversation, the trading partner can leave the conversation by selecting `Leave this Conversation`.

Monitoring Collaboration Agreements

A collaboration agreement specifies the trading partners, delivery channels, and roles that define a specific interaction. You can view collaboration agreement status by selecting a collaboration agreement from the navigation tree or the Collaboration Agreements page, and then selecting the Monitoring tab.

The status of a collaboration agreement is displayed as shown in the following figure.

Figure 5-6 Monitoring a Collaboration Agreement



The information displayed includes the following: identifying information, status information, the business protocol, the conversation definition, and the number of parties.

For successful deployment, a collaboration agreement must include two parties, and both of the delivery channel endpoints assigned in the collaboration agreement must be bound to the same business protocol. (Protocol binding is assigned in the document exchange assigned to a delivery channel.)

If a collaboration agreement specifies a spoke delivery channel for the local trading partner, the agreement cannot be deployed successfully until the hub delivery channel is up and running.

You always have the option of reversing the status of a collaboration agreement:

- You can disable an enabled collaboration agreement by selecting `Disable this Collaboration Agreement`.
- You can enable a disabled collaboration agreement by selecting `Enable this Collaboration Agreement`.

When you disable a collaboration agreement, existing conversations are completed, but no new conversations for that agreement can be started, and no new participants can be added to existing conversations.

Similarly, you can register an unregistered collaboration agreement or unregister a registered collaboration agreement by selecting `Register this Collaboration Agreement` or `Unregister this Collaboration Agreement`, respectively. Unregistering a collaboration agreement triggers termination of all active conversations associated with the collaboration agreement.

You can also view the status of collaboration agreements for a delivery channel, as follows:

1. Display a list of delivery channels by performing the steps described in “Monitoring Delivery Channels” on page 5-11.
2. Select a delivery channel from the list. The number of collaboration agreements for that delivery channel is displayed in the summary statistics for the delivery channel.
3. View a list of the collaboration agreements for that delivery channel by clicking the link in the number of collaboration agreements.
4. From the list, select a collaboration agreement for which you want to see status information like the information shown in the following figure.

Figure 5-7 Monitoring a Collaboration Agreement



In this case, you cannot disable or unregister the collaboration agreement.

Monitoring Messages

To display a list of the outstanding messages for a trading partner session, complete the following procedure:

1. Display a list of trading partner sessions. (For instructions, see “Monitoring Trading Partner Sessions” on page 5-9.)
2. Select a trading partner session from the list.

When you make your selection, the number of outstanding messages for the selected trading partner session is displayed in the summary statistics for that session.

3. To display a list of outstanding messages for the designated trading partner session, click the number (in the summary statistics) representing the total number of outstanding messages.

When you select a message from a list, the following information is displayed: identifying information, the date and time at which the message was sent, and the size of the message.

6 Working with the Repository

The following sections describe the WebLogic Integration repository:

- Understanding the Repository
- Managing the B2B Configuration Information in the Repository

Understanding the Repository

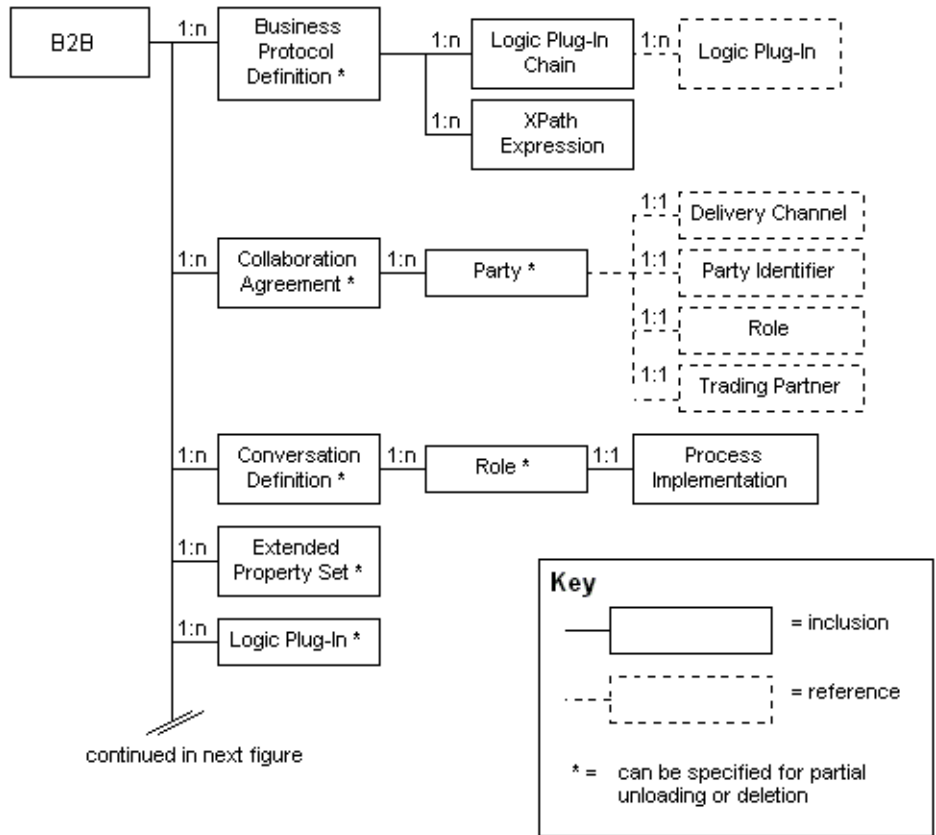
The WebLogic Integration repository is the database that stores the information required by WebLogic Integration. The repository consists of:

- B2B configuration tables—set of database tables that contain the static B2B configuration information.
- Persistence tables—set of database tables in which the persistence state is stored. (dynamic)
- JMS Queue tables—set of database tables that store the message state. (dynamic)
- Workflow tables—set of database tables that store static workflows and dynamic run-time information.
- Common tables—set of database tables that store XML, DTDs, XLST style sheets, and other entities used by WebLogic Integration.

B2B Configuration Elements

The following two figures show the relationships among the B2B configuration elements in the repository. For information about configuring the repository data elements, see Chapter 1, “Configuration Requirements.”

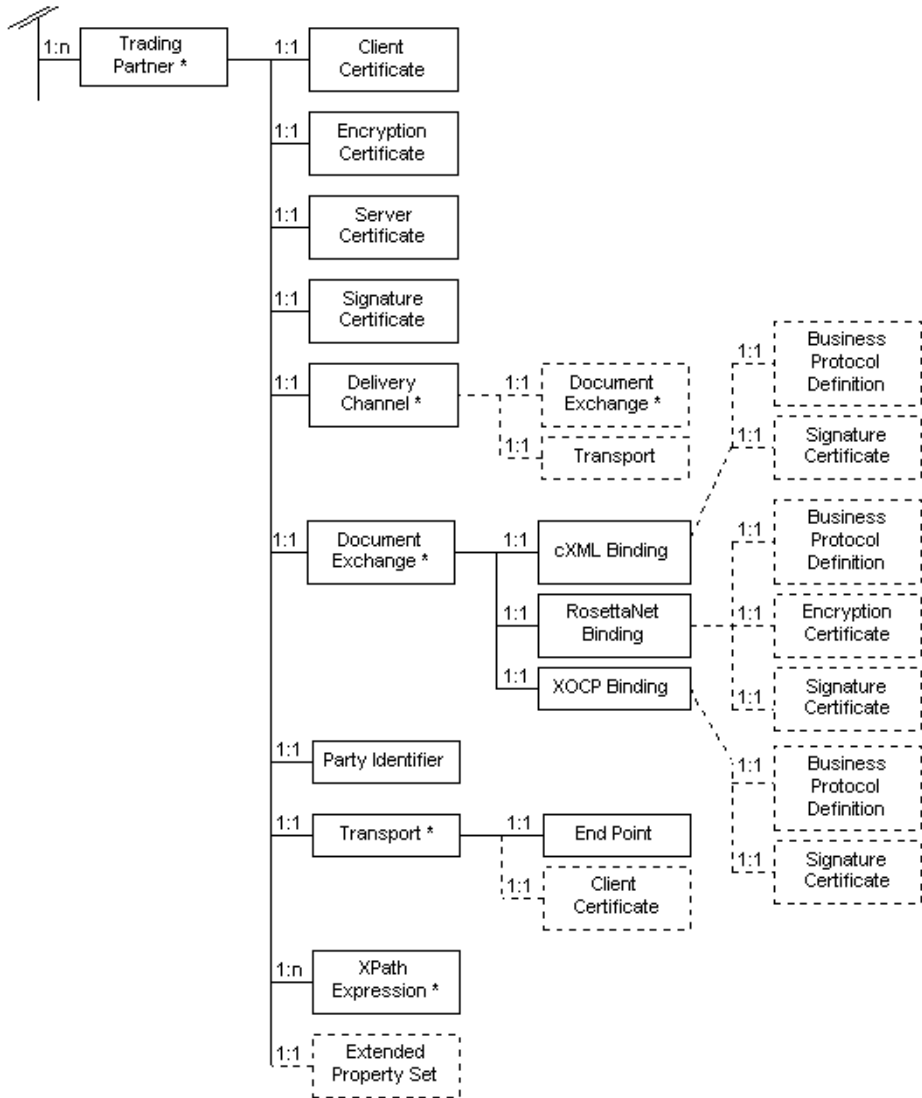
Figure 6-1 Elements in the Repository (Part 1)



The following figure is a continuation of the previous figure.

Figure 6-2 Elements in the Repository (Part 2)

continued from previous figure



In the figures, solid lines represent inclusion and dashed lines represent reference. When you remove an element from the repository, the following events occur:

- If the removed element includes other elements, the included elements are removed from the repository, too.
- If the removed element refers to other elements, the referenced elements remain in the repository.

For example, if you remove a transport, the transport's end point is also removed but the referenced client certificate remains.

The following table provides an overview of the elements in the repository.

Table 6-1 Elements in the Repository

Element	Description
B2B	<p>The B2B root element. This element represents the B2B engine configuration. All major elements stem from this root element.</p> <p>Note: The B2B root element is <code>w1c</code>. This abbreviation is a legacy from a previous release.</p>
Business protocol definition	<p>Business protocol definition that specifies how the B2B engine processes business messages, including how it reads messages and how it routes messages to recipients. A business protocol definition also specifies persistence, retries, and quality of service.</p>
Logic plug-in chain	<p>A logic plug-in chain is a set of logic plug-ins that changes the way in which B2B integration routes or filters a message.</p>
XPath expression	<p>An XPath expression is a string written in XPath syntax that, when evaluated, results in one of the following types of objects:</p> <ul style="list-style-type: none">■ Node-set (an unordered collection of nodes without duplicates)■ Boolean■ Floating-point number■ String <p>For information about XPath expressions, see “Expressions” at the following URL:</p> <p>http://www.w3.org/TR/xpath.html#section-Expressions</p>

Table 6-1 Elements in the Repository (Continued)

Element	Description
Collaboration agreement	A collaboration agreement is a definition of the interactions that trading partners agree to carry out, along with a specification for the methods through which these interactions are conducted. This specification includes details about transport, messaging, security constraints, and bindings to a process specification.
Party	A party is an entity that binds a role in a conversation definition to a trading partner in a collaboration agreement.
Conversation definition	A conversation definition is a collection of values that defines a conversation.
Role	A role is a definition of activities, such as buying and selling, that can be performed by a trading partner during a conversation. A role is defined in terms of the documents that can be sent or received by a trading partner in the conversation. Each conversation has two or more roles, and each role is defined by a collaborative workflow.
Process implementation	Associates the workflow template name and organization.
Extended property set	An extended property set is a set of user-defined elements, attributes, or text components that can be associated with entities in the repository.
Logic plug-in	<p>Logic plug-ins are Java classes that are invoked when WebLogic Integration is started. At run time, logic plug-ins intercept, process, and output business messages. Built-in router and filter logic plug-ins are associated with each supported business protocol. Customer-provided logic plug-ins can modify the default routing and filtering functionality, or provide functionality other than routing and filtering, such as billing.</p> <p>For information about logic plug-ins, see Introducing B2B Integration and Programming Logic Plug-Ins for B2B Integration.</p>
Trading partner	A trading partner is a business entity that is authorized to send and receive business messages in a conversation.
Certificates	<p>WebLogic Integration supports the following types of digital certificates for use in B2B integration:</p> <ul style="list-style-type: none"> ■ Client certificates ■ Encryption certificates ■ Server certificates ■ Signature certificates

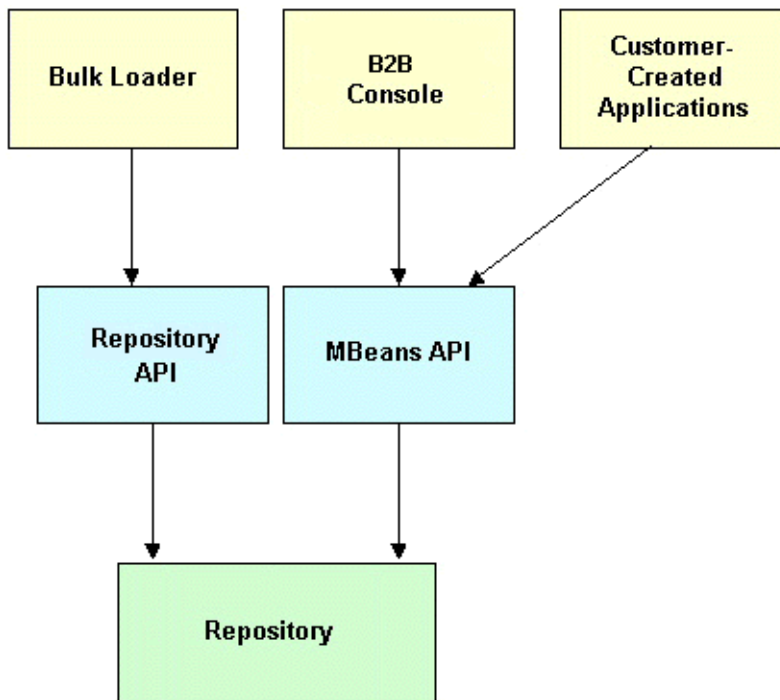
Table 6-1 Elements in the Repository (Continued)

Element	Description
Delivery channel	A delivery channel is a specification for delivering business messages to one trading partner. There is one delivery channel per trading partner per collaboration agreement.
Document exchange	A document exchange is a definition of the method through which a document is exchanged. A document exchange defines a business protocol and some run-time parameters.
Bindings for business protocols	To participate in a business process, a trading partner needs to define a binding for the business protocol that the business process uses. The binding associates the protocol with a specific delivery channel.
Party identifier	A party identifier is a value that specifies a trading partner participant to a conversation.
Transport	A transport specifies the transport level properties for a delivery channel.
Endpoint	An endpoint is the URI defined for the delivery channel.

Managing the B2B Configuration Information in the Repository

The following figure illustrates the methods you can use to manage the information in the repository.

Figure 6-3 Managing the Information in the Repository



As this figure illustrates, when you update information through the B2B Console or a custom management application, the repository is accessed by means of the MBeans API. The Bulk Loader employs a low-level API to handle database operations.

The MBeans API enables you to update the repository dynamically at run time. That is, database updates can occur while the B2B engine is running. The Bulk Loader, on the other hand, requires B2B engine shutdown.

Additional information about each of these methods of updating the B2B configuration elements stored in the repository can be found in the following documents:

- For information about using the Bulk Loader, see Chapter 7, “Working with the Bulk Loader.”
- For information about using the B2B console, see the *Online Help for the WebLogic Integration B2B Console*.
- For information about creating your own applications, see the WebLogic Integration development guides.

7 Working with the Bulk Loader

The following sections describe how to work with the Bulk Loader:

- Understanding the Terminology
- Importing Data into the Repository
- Exporting Data from the Repository
- Deleting Data from the Repository
- Working with the Bulk Loader Configuration File
- Working with the Repository Data File
- Checking Data
- Forcing the Bulk Loader

In addition to using the Bulk Loader, you can use the B2B Console to transfer data to and from the repository, as described in Chapter 4, “Importing and Exporting B2B Integration Components.” For information about the repository, see Chapter 6, “Working with the Repository.”

Understanding the Terminology

The following table describes some of the terms used in this chapter.

Table 7-1 Terms Used in This Chapter

This term . . .	Refers to . . .
XML declaration	A processing instruction that identifies the document as being XML. At a minimum, the XML version must be specified. If the document uses a character set other than the default, <code>encoding="valid_encoding"</code> must also be specified.
Doctype declaration	The XML document type (references the root element and DTD).
XML element	An element in an XML file.
attribute	An attribute for an XML element.
value	The value of either an attribute or a data element.

For example, consider the following repository data file:

```
<?xml version="1.0"?>
<!DOCTYPE wlc SYSTEM "WLC.dtd">
<wlc system-password="wlcsystem">
  <conversation-definition
    name="CMQPAConversation"
    version="1.1"
    business-protocol-name="XOCP"
    protocol-version="1.1">
    <role
      name="CMBuyer"
      wlpi-template="CMBuyerQPAPublic">
      <process-implementation wlpi-org="ORG1" />
    </role>
    <role
      name="CMSupplier"
      wlpi-template="CMSupplierQPAPublic">
      <process-implementation wlpi-org="ORG1" />
    </role>
  </conversation-definition>
</wlc>
```

In the previous file, the following examples of the definitions are used:

- `<?xml version="1.0"?>` is the XML declaration.
- `conversation-definition` is an XML element. The child elements and attributes of the `conversation-definition` element constitute the conversation definition stored in the repository.
- `name` is an attribute of the `conversation-definition` element.
- `"CMQPAConversation"` is the value for the `name` attribute.

Importing Data into the Repository

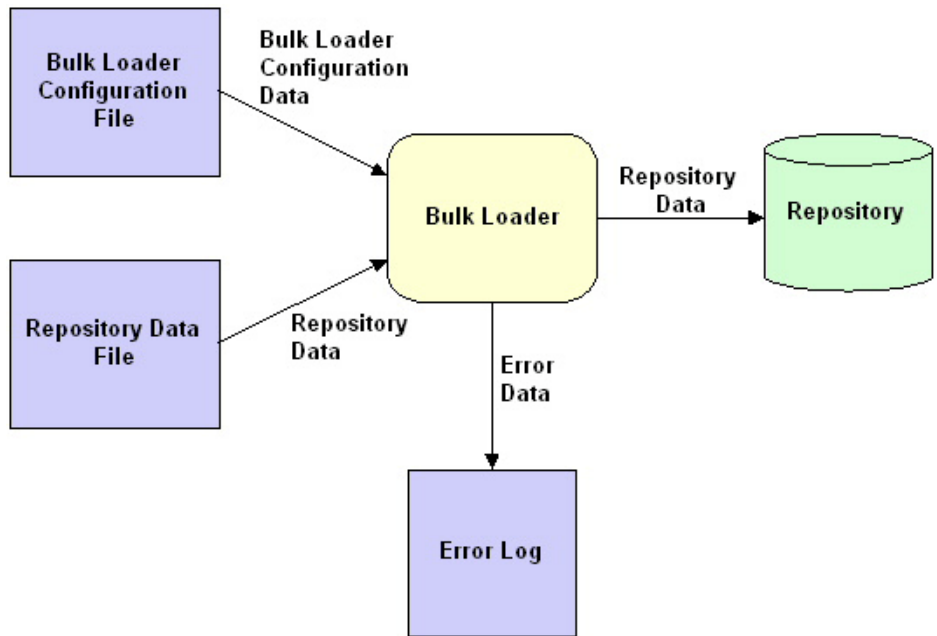
The following sections provide information about importing repository data:

- How the Bulk Loader Imports Data
- Procedure for Importing Data into the Repository

How the Bulk Loader Imports Data

As the following figure shows, the Bulk Loader parses a Bulk Loader configuration file to get instructions about transferring data from a repository data file into the repository. Both files are XML files. If the Bulk Loader detects any errors during this procedure, it creates an error log.

Figure 7-1 Using the Bulk Loader to Import Repository Data



The Bulk Loader uses the following logic to import data from a repository data file into the repository:

1. If the B2B root element specified in the repository data file does not exist in the repository, the Bulk Loader creates it.

Note: The B2B root element in a repository data XML file is `w1c`. This abbreviation is a legacy from a previous release.

2. If the B2B root element specified in the repository data file already exists in the repository, the Bulk Loader uses the following logic to process each data element in the repository data file:
 - If the data element exists in the repository and has the same data element values, attributes, and attribute values as the corresponding XML element in the repository data file, the Bulk Loader does nothing.
 - If the data element does not exist in the repository, the Bulk Loader creates it, using the logic described in the following table for each data element value and attribute.
 - If the data element exists in the repository but includes one or more data element values, attributes, or attribute values that do not match the values of the corresponding XML element in the repository data file, the Bulk Loader recreates the data element using the logic described in the following table for each data element value and attribute.

The following table describes the logic the Bulk Loader uses to recreate an existing element.

Table 7-2 Logic for Processing an Attribute

Does the element or attribute exist in the repository data file?	Attribute type	Logic
Yes	Does not matter	The Bulk Loader sets the data element value or attribute to the value specified in the repository data file.
No	IMPLIED	The Bulk Loader sets the data element value or attribute to null unless it is one of the attributes that has a special default value, as described in the following table.
No	REQUIRED	The Bulk Loader considers the data to be invalid. For more information, see “Checking Data” on page 7-23.

The following table lists the default values for special IMPLIED attributes.

Table 7-3 Default Values for Special IMPLIED Attributes

XML Element	Attribute	Default Value
wlc	large-msg-support-on	OFF
wlc	show-hidden	OFF
trading-partner	status	ENABLED

Note: The abbreviation `wlc` is a legacy from a previous release of the product. The `wlc` element corresponds to the top-level node, `B2B`, in the B2B Console.

Procedure for Importing Data into the Repository

Notes: In addition to using the Bulk Loader, you can use the B2B Console to transfer data to and from the repository, as described in Chapter 4, “Importing and Exporting B2B Integration Components.”

You cannot run the Bulk Loader when the B2B engine is running. If the B2B engine has shut down abnormally, thus preventing the Bulk Loader from running, you can force the import, as described in “Forcing the Bulk Loader” on page 7-26.

To import data from a repository data file into the repository:

1. Create a Bulk Loader configuration file.

In it, include `load-processing-parameters`, which is the XML element that instructs the Bulk Loader to import data from the repository data file into the repository. For information about creating a Bulk Loader configuration file, see “Working with the Bulk Loader Configuration File” on page 7-14.

2. Create a repository data file.

For information about creating a repository data file, see “Working with the Repository Data File” on page 7-21.

3. To import data from the repository data file into the repository, enter the command appropriate for your platform:

- Windows: `bulkloader cfg_file`
- UNIX: `bulkloader.sh cfg_file`

In both commands, *cfg_file* is the pathname of the Bulk Loader configuration file that you created in step 1. The Bulk Loader configuration file, in turn, specifies the pathname of the repository data file that you created in step 2.

Use the same command for importing, exporting, and deleting data. The Bulk Loader configuration file indicates which action the Bulk Loader should take.

While importing data, the Bulk Loader checks for errors, as described in “Checking Data” on page 7-23.

Exporting Data from the Repository

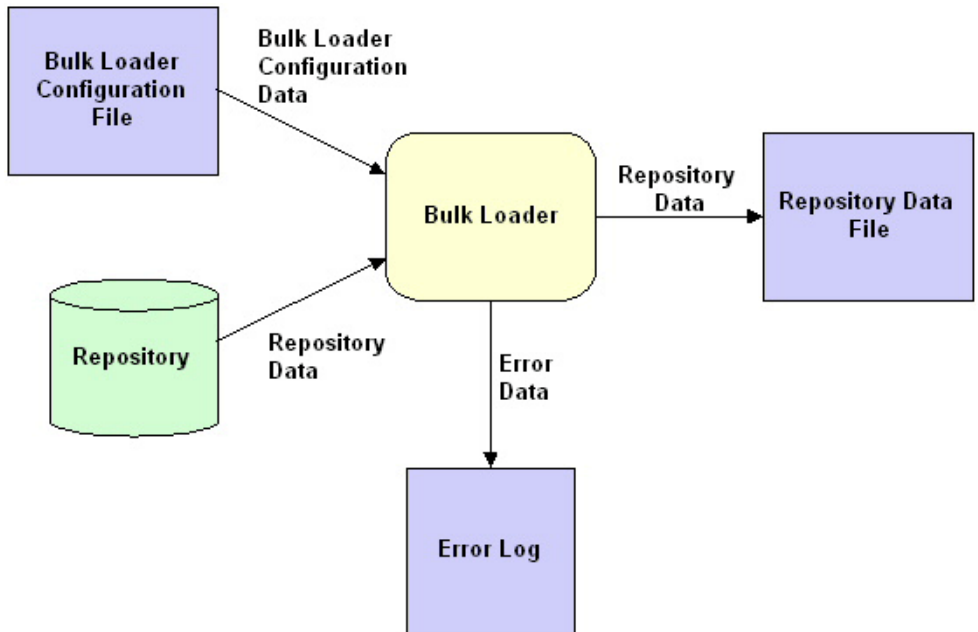
The following sections provide information about exporting repository data:

- How the Bulk Loader Exports Data
- Full and Partial Repository Exports
- Short and Long Repository Exports
- Procedure for Exporting Repository Data

How the Bulk Loader Exports Data

As the following figure shows, the Bulk Loader parses a Bulk Loader configuration file to get instructions about transferring data from the repository to a repository data file. Both files are XML files. If the Bulk Loader detects any errors during this procedure, it creates an error log.

Figure 7-2 Using the Bulk Loader to Export Repository Data



Full and Partial Repository Exports

When you export data from the repository, you can specify a full export or a partial export, as described in the following table.

Table 7-4 Full and Partial Repository Exports

Type of Export	Description
Full	All the data is exported.
Partial	A subset of the data is exported.

By default, the Bulk Loader performs a full repository export.

To perform a partial repository export, use the `entities` XML element in the Bulk Loader configuration file. The `entities` XML element specifies the data elements to be exported. The Bulk Loader uses the `entities` values to traverse the repository to the specified data elements.

For a complete description of the `entities` XML element, see the `WLCCConfig.dtd` file in the `lib\dtd` subdirectory of your WebLogic Integration installation directory.

You can identify a specific data element instance in the Bulk Loader configuration file. For example, to export a specific collaboration agreement, include the `<entities>` element and identifying information for the collaboration agreement in the required structure, as shown in the following listing. This strategy applies to all types of data elements.

Listing 7-1 Example of a Bulk Loader Configuration File for Exporting a Specific Collaboration Agreement

```
<?xml version="1.0"?>
<!DOCTYPE wlc-config SYSTEM "WLCConfig.dtd">
<wlc-config>
  <unload-processing-parameters>
    <database-url>jdbc:weblogic:oracle:REPO</database-url>
    <database-driver>weblogic.jdbc.oci.Driver</database-driver>
    <database-user-id>scott</database-user-id>
    <database-password>tiger</database-password>
    <xml-file-name>ExportRepoData.xml</xml-file-name>
    <entities>
      <wlc>
        <collaboration-agreement
          name="QPA-XOCP-TP1-TP2"
          version=1.2>
        </collaboration-agreement>
      </wlc>
    </entities>
  </unload-processing-parameters>
</wlc-config>
```

Short and Long Repository Exports

When you export data from the repository, you can specify a short export or a long export, as described in the following table.

Table 7-5 Short and Long Repository Exports

For this type of export . . .	The Bulk Loader organizes the output data . . .
Short (standard)	To represent the user's view of the repository.
Long (extensive)	As a snapshot of the repository, which can be useful when you migrate repository data from one database to another.

By default, the Bulk Loader exports repository data in the short format.

To perform a long repository export, set the `format` attribute for the `unload-processing-parameters` XML element in the Bulk Loader configuration file to `long`.

Because the long format includes internal repository data in addition to values for various objects, we recommend that you use the long format only for the following reasons:

- To save a backup of the entire repository. For example, before you delete data from the repository, it is a good idea to back up the repository.
- To migrate repository data from one environment to another. For example, if you change from one database vendor to another, or if you upgrade your database to a new machine, then you need to use the long format to migrate the entire repository database.

For an example of a Bulk Loader configuration file that specifies a `format` value, see Listing 7-3. For a complete description of the `format` attribute, see the `WLCConfig.dtd` file in the `lib\dtd` subdirectory of your WebLogic Integration installation directory, and “Short and Long Repository Exports” on page 7-10.

Procedure for Exporting Repository Data

Note: In addition to using the Bulk Loader, you can use the B2B Console to transfer data to and from the repository, as described in Chapter 4, “Importing and Exporting B2B Integration Components.”

You cannot run the Bulk Loader when the B2B engine is running. If the B2B engine has shut down abnormally, thus preventing the Bulk Loader from running, you can force the export, as described in “Forcing the Bulk Loader” on page 7-26.

To export data from the repository to a repository data file:

1. Create a Bulk Loader configuration file.

In it, include `unload-processing-parameters`, which is the XML element that instructs the Bulk Loader to export data from the repository to a repository data file. For information about creating a Bulk Loader configuration file, see “Working with the Bulk Loader Configuration File” on page 7-14.

2. To export data from the repository to a repository data file, enter the command appropriate for your platform:

- Windows: `bulkloader cfg_file`
- UNIX: `bulkloader.sh cfg_file`

In both commands, *cfg_file* is the pathname of the Bulk Loader configuration file that you created in step 1. The Bulk Loader configuration file, in turn, specifies the pathname of the repository data file into which the Bulk Loader exports the data.

Use the same command for importing, exporting, and deleting data. The Bulk Loader configuration file indicates which action the Bulk Loader should take.

While exporting data, the Bulk Loader checks for errors, as described in “Checking Data” on page 7-23.

Deleting Data from the Repository

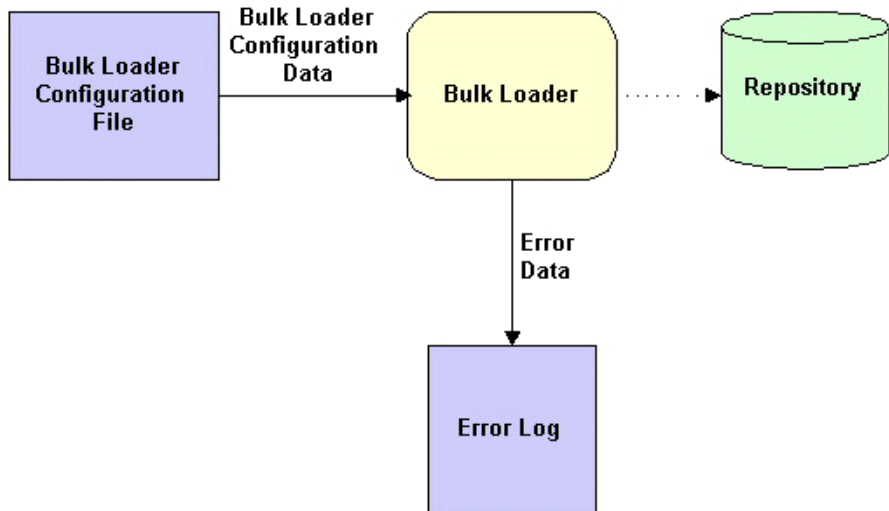
The following sections provide information about deleting repository data:

- How the Bulk Loader Deletes Data
- Procedure for Deleting Repository Data

How the Bulk Loader Deletes Data

As the following figure shows, the Bulk Loader parses a Bulk Loader configuration file, which is an XML file, to get instructions about deleting data from the repository. The dotted line in the figure indicates that the Bulk Loader affects the repository without sending any data to it. If the Bulk Loader detects any errors during this procedure, it creates an error log.

Figure 7-3 Using the Bulk Loader to Delete Repository Data



Procedure for Deleting Repository Data

Note: You cannot run the Bulk Loader when the B2B engine is running. If the B2B engine has shut down abnormally, thus preventing the Bulk Loader from running, you can force the deletion of the data, as described in “Forcing the Bulk Loader” on page 7-26.

To delete repository data:

1. We strongly recommend that you back up the repository before deleting any data from it. To back up the repository, perform a full, long (extensive) export, as described in “Exporting Data from the Repository” on page 7-7.
2. Create a Bulk Loader configuration file.

In it, include `delete-processing-parameters`, which is the XML element that instructs the Bulk Loader to delete data from the repository. For information about creating a Bulk Loader configuration file, see “Working with the Bulk Loader Configuration File” on page 7-14.

3. To delete data from the repository, enter the command appropriate for your platform:

- Windows: `bulkloader cfg_file`
- UNIX: `bulkloader.sh cfg_file`

In both commands, `cfg_file` is the pathname of the Bulk Loader configuration file that you created in step 2.

Use the same command for importing, exporting, and deleting data. The Bulk Loader configuration file indicates which action the Bulk Loader should take.

While deleting data, the Bulk Loader checks for errors, as described in “Checking Data” on page 7-23.

Working with the Bulk Loader Configuration File

Bulk Loader configuration files contain the database login information and processing instructions required to carry out the desired task. Each Bulk Loader configuration is an XML file that conforms to the `WLCCConfig.dtd`.

Although the sections that follow provide much of the information you need to create your own Bulk Loader configuration file, you may need to refer to the `WLCCConfig.dtd` located in the `lib\dtd` subdirectory for certain details.

A Bulk Loader configuration file has a fairly simple structure. The root element `<wlc-config>` contains one of the following elements, depending on the action:

- When importing, `<wlc-config>` contains the `<load-processing-parameters>` element. The processing parameters used to import the data are attributes or direct children of `<unload-processing-parameters>`.
- When exporting, `<wlc-config>` contains the `<unload-processing-parameters>` element. The processing parameters used to export the data are attributes or direct children of `<unload-processing-parameters>`.

- When exporting, `<wlc-config>` contains the `<delete-processing-parameters>` element. The processing parameters used to delete the data are attributes or direct children of `<delete-processing-parameters>`.

The following table summarizes the processing parameters that can be specified to carry out each type of action.

Table 7-6 Bulk Loader Configuration Processing Parameters

Elements and Attributes	Description
Attribute: <code>transaction-level</code> Is an attribute of: <code><load-processing-parameters></code> <code><delete-processing-parameters></code>	Specifies whether the data is imported in a single transaction, or in multiple transactions. Valid values are <code>all</code> or <code>default</code> . When set to <code>default</code> , or not specified, a transaction is initiated upon import or delete of each of the following entities: trading partners, conversation definitions, collaboration agreements, business protocol definitions, and logic plug-ins. If invalid data is detected during a transaction for any entity, the import is rolled back for the current transaction only; importing continues for the next transaction. For additional information about error detection, see “Checking Data Integrity” on page 7-24.
Attribute: <code>database-initialization</code> Is an attribute of: <code><load-processing-parameters></code>	Specifies whether or not to initialize the database. Valid values are <code>yes</code> and <code>no</code> . <code>yes</code> —clears the existing B2B integration data from the repository during the import. <code>no</code> —retains existing B2B integration data during the import. If the imported data describes elements and attributes that already exist in the repository, then the imported data overwrites the existing data for those specific values. (<code>no</code> is the default)
Attribute: <code>format</code> Is an attribute of: <code><unload-processing-parameters></code>	Specifies whether the Bulk Loader exports data in the long (extensive) format or in the short (standard) format. Valid values are <code>short</code> and <code>long</code> . For more information, see “Short and Long Repository Exports” on page 7-10.

Table 7-6 Bulk Loader Configuration Processing Parameters (Continued)

Elements and Attributes	Description
<p>Attribute: <database-url></p> <p>Is a required child of: <load-processing-parameters> <unload-processing-parameters> <delete-processing-parameters></p>	<p>URL for the database, as specified in the JDBC driver documentation. For a summary of the required URLs for supported drivers, see “Database Access Parameters” in “Customizing WebLogic Integration” in <i>Starting, Stopping, and Customizing BEA WebLogic Integration</i></p>
<p>Element: <database-driver></p> <p>Is a required child of: <load-processing-parameters> <unload-processing-parameters> <delete-processing-parameters></p>	<p>JDBC driver required to connect to the database. For supported drivers, see “Database Access Parameters” in “Customizing WebLogic Integration” in <i>Starting, Stopping, and Customizing BEA WebLogic Integration</i></p>
<p>Element: <database-user-id></p> <p>Is an optional child of: <load-processing-parameters> <unload-processing-parameters> <delete-processing-parameters></p>	<p>Database user ID, if required to connect to the database.</p>
<p>Element: <database-password></p> <p>Is an optional child of: <load-processing-parameters> <unload-processing-parameters> <delete-processing-parameters></p>	<p>Database password, if required to connect to the database.</p>
<p>Element: <database-properties></p> <p>Is an optional child of: <load-processing-parameters> <unload-processing-parameters></p>	<p>Allows you to specify any properties the database driver may require, using the following syntax: <code>name1=value1[;name2=value2;name3=value3 . . .]</code></p> <p>In particular, if you are using a WebLogic jDriver, and the database is configured to use an alternate character set, then you must set the <code>weblogic.codeset</code> property. For example, if you are using the <code>Shift_JIS</code> character set in your locale, specify: <code>weblogic.codeset=Shift_JIS</code></p>

Table 7-6 Bulk Loader Configuration Processing Parameters (Continued)

Elements and Attributes	Description
Element: <code><xml-file-name></code> Is a required child of: <code><load-processing-parameters></code> <code><unload-processing-parameters></code>	Contains the name of the existing XML file to be imported, or the new XML file to be created to store the exported data. Unless the file is located in the current directory, the full path must be specified.
Element: <code><encoding></code> Is an optional child of: <code><unload-processing-parameters></code>	Used to export the B2B configuration elements using the specified encoding (character set/code set). For a list of valid character set names and aliases, see http://www.iana.org/assignments/character-sets
Element: <code><entities></code> Is an optional child of: <code><unload-processing-parameters></code> Is a required child of: <code><delete-processing-parameters></code>	Contains the entities to be deleted or unloaded. Refer to <code>WLCConfig.dtd</code> for definitions of the entity elements and subelements.

To use the Bulk Loader to import, export, or delete data from the repository:

1. Create an XML file that specifies `WLCConfig.dtd` in the DocType declaration, and conforms to the structure required by the DTD.
2. If you are importing or exporting data, set the `xml-file-name` XML element in the XML file to specify the pathname of the repository data file. If you specify only a filename, the Bulk Loader looks for the repository data file in the current working directory.

The following sections provide example Bulk Loader configuration files for importing and exporting data:

- Bulk Loader Configuration File for Importing Data
- Bulk Loader Configuration File for Exporting Data

Bulk Loader Configuration File for Importing Data

The following listing is an example Bulk Loader configuration file for importing data into the repository.

Listing 7-2 Bulk Loader Configuration File—Import

```
<?xml version="1.0"?>
<!DOCTYPE wlc-config SYSTEM "WLCConfig.dtd">
<wlc-config>
  <load-processing-parameters database-initialization="no" \
    transaction-level="all">
    <database-url>
      jdbc:oracle:thin:@rdbmshost:1521:WLIDB
    </database-url>
    <database-driver>
      oracle.jdbc.driver.OracleDriver
    </database-driver>
    <database-user-id>
      scott
    </database-user-id>
    <database-password>
      tiger
    </database-password>
    <xml-file-name>
      ImportRepoData.xml
    </xml-file-name>
  </load-processing-parameters>
</wlc-config>
```

In addition to the processing parameters that are required in any Bulk Loader configuration file (DTD, database URL, database driver, database user ID, and database password), this example defines the following:

- `database-initialization`—The value for this attribute specifies whether the Bulk Loader deletes all data from the repository before performing the repository import. Valid values are `yes` and `no`.
- `transaction-level`—The value for this attribute specifies the actions for the Bulk Loader to take if it detects an error. Valid values are `all` and `default`. For more information, see “Checking Data Integrity” on page 7-24.

- `xml-file-name`—The value for this element specifies the pathname of the repository data file from which the Bulk Loader imports the data. If you specify only a filename, the Bulk Loader looks for the repository data file in the current working directory.

Although not shown in this example, the `database-properties` element is also available when exporting. This element can be used to specify any additional properties required by the JDBC driver.

Bulk Loader Configuration File for Exporting Data

The following listing is an example Bulk Loader configuration file for exporting data from the repository. In this case, the configuration file itself contains locale-specific characters (see `xml-file-name`), and therefore, the XML declaration must specify the appropriate encoding.

Listing 7-3 Bulk Loader Configuration File—Export

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE wlc-config SYSTEM "WLCConfig.dtd">
<wlc-config>
  <unload-processing-parameters format="short">
    <database-url>
      jdbc:weblogic:mssqlserver4:WLIDB@myhost
    </database-url>
    <database-driver>
      weblogic.jdbc.mssqlserver4.Driver
    </database-driver>
    <database-user-id>
      myuserid
    </database-user-id>
    <database-password>
      mypassword
    </database-password>
    <xml-file-name>
      d:\beal\RN2AITest\@nA&\exportedPeer1config.xml
    </xml-file-name>
    <encoding>
      GB2312
    </encoding>
    <database-properties>
      weblogic.codeset=GB2312
    </database-properties>
  </unload-processing-parameters>
</wlc-config>
```

In addition to specifying values that are required in any Bulk Loader configuration file (DTD, database URL, database driver, database user ID, and database password), this example defines the following:

- `XML` declaration specifies encoding—Because the file itself includes characters from the GB2312 character set (Chinese for People’s Republic of China, mixed one-byte, two-byte set) the XML declaration sets `encoding=GB2312`.
- `format`—The value for this attribute specifies whether the Bulk Loader exports data in the long (extensive) format or in the short (standard) format. Valid values are `short` and `long`. For more information, see “Short and Long Repository Exports” on page 7-10.
- `xml-file-name`—The value for this element specifies the pathname of the repository data file to which the Bulk Loader exports the data. If you specify only a filename, the Bulk Loader looks for the repository data file in the current working directory.
- `encoding`—The value for this element specifies the encoding to be used in the output file.
- `database-properties`—In order to connect properly to the Microsoft SQL database for the locale, the `weblogic.codeset` property must be set to the encoding used in the database.

Working with the Repository Data File

The repository data file is an XML file that conforms to the `wlc.dtd`. To create a repository data file, create an XML file that specifies `wlc.dtd` in the DocType declaration, and conforms to the structure required by the DTD. Refer to the `wlc.dtd`, which is in the `lib\dtd` subdirectory of your WebLogic Integration installation directory, for the required structure.

The following listing shows a repository data file that creates an extended property set.

Listing 7-4 Example of a Repository Data File for a New Extended Property Set

```
<?xml version="1.0"?>
<!DOCTYPE wlc SYSTEM "WLC.dtd">
<wlc
  name="WLC"
  large-msg-support-on="ON"
  large-msg-min-size="10000"
  large-msg-location="c:\temp"
  proxy-host="andrew"
  proxy-port="7502"
  description="The WLC Hub" >

  <extended-property-set name="EPS 1">
    <xml-element name="C1">
      <xml-attribute name="C1_A1" value="C1_A1 Value"/>
      <xml-element name="C1G1">
        <xml-element name="C1G1_T1" text="C1G1_T1 Value"></xml-element>
        <xml-element name="C1G1_T2" text="C1G1_T2 Value"></xml-element>
        <xml-element name="C1G1_T3" text="C1G1_T3 Value"></xml-element>
      </xml-element>
      <xml-element name="C1G2" text="C1G2 Value"></xml-element>
      <xml-element name="C1G3" text="C1G3 Value"></xml-element>
    </xml-element>
    <xml-element name="C2">
      <xml-attribute name="C2_A1" value="C2_A1 Value"/>
      <xml-attribute name="C2_A2" value="C2_A2 Value"/>
      <xml-attribute name="C2_A3" value="C2_A3 Value"/>
      <xml-element name="C2G1" text="C2G1 Value"></xml-element>
      <xml-element name="C2G2">
        <xml-element name="C2G2_T1" text="C2G2_T1 Value"></xml-element>
        <xml-element name="C2G2_T2" text="C2G2_T2 Value"></xml-element>
        <xml-element name="C2G2GG1">
          <xml-attribute name="C2G2GG1_A1" value="C2G2GG1_A1 Value"/>
          <xml-element name="C2G2GG1_T1" text="C2G2GG1_T1
            Value"></xml-element>
        </xml-element>
      </xml-element>
      <xml-element name="C2G3" text="C2G3 Value"></xml-element>
    </xml-element>
  </extended-property-set>
</wlc>
```

Checking Data

The following sections describe how the Bulk Loader checks data:

- Creating an Error Log
- Validating XML Files
- Checking Data Integrity

Creating an Error Log

To keep track of errors, the Bulk Loader creates a file named `wlc.log` in the current working directory. If a `wlc.log` file already exists, the Bulk Loader renames the existing file by appending a timestamp to the name in the following format:

`wlc.log.yyyy.mm.dd.hh.mi.ss`. Then the Bulk Loader creates a new `wlc.log` file. The following table describes the fields in the timestamp.

Table 7-7 Timestamp Appended to Name of Existing Log File

This string . . .	Indicates the . . .
<i>yyyy</i>	Year
<i>mm</i>	Month in numeric format (between 01 - 12)
<i>dd</i>	Day in numeric format
<i>hh</i>	Hour (between 00 - 23)
<i>mi</i>	Minute
<i>ss</i>	Second

Validating XML Files

Before it processes any data, the Bulk Loader validates the XML files. The following table lists the files that the Bulk Loader validates for each type of Bulk Loader task.

Table 7-8 Files Validated by the Bulk Loader

For this task . . .	The Bulk Loader validates . . .
Importing data	Bulk Loader configuration file
	Repository data file
Exporting data	Bulk Loader configuration file
Deleting data	Bulk Loader configuration file

To validate one of these types of XML files, the Bulk Loader checks it against the corresponding `.dtd` file. If the Bulk Loader detects an error in the XML file, it stops without processing the data.

Checking Data Integrity

After validating the XML files, the Bulk Loader checks the data integrity while it is processing the data. To check data integrity, the Bulk Loader verifies that the information in the XML files does not conflict with the information in the repository.

For example, if the Bulk Loader is adding a new data element to the repository and if the new data element references another data element, the Bulk Loader makes sure that the referenced data element exists in the repository or in the repository data file.

Checking Data Integrity While Importing or Deleting

To have data integrity checked while data is being imported or deleted, set the `transaction-level` attribute in the Bulk Loader configuration file, as shown in Listing 7-3. If you do not set `transaction-level`, the Bulk Loader uses the `default` value. The Bulk Loader performs one of the following sets of actions, depending on the value of `transaction-level`:

- `all`—The Bulk Loader performs a single transaction for all the data. If the Bulk Loader detects invalid data during the transaction, it rolls back the entire transaction and stops. The repository is left in exactly the same condition it was in before the Bulk Loader started importing or deleting data.
- `default`—The Bulk Loader performs a separate transaction for each of the following types of data elements:
 - Extended property set
 - Text document
 - Message definition
 - Logic plug-in
 - Business process
 - Role
 - Business protocol definition
 - Trading partner profile
 - Document exchange
 - Transport
 - Delivery channel
 - XPath expression
 - Collaboration protocol agreement
 - Party

If the Bulk Loader detects invalid data during a transaction for one of these types of data elements, it rolls back the current transaction and then performs the next transaction.

If the Bulk Loader detects invalid data for a B2B configuration data element at the B2B level (such as a `wlc` attribute), it rolls back all transactions that have been performed for that B2B element and stops.

Checking Data Integrity During an Export

If the Bulk Loader detects invalid data in the Bulk Loader configuration file, it does not perform the export.

Forcing the Bulk Loader

By default, the Bulk Loader does not import, export, or delete data while the B2B engine is running. If, for some reason, the B2B engine does not shut down normally, the Bulk Loader fails with the following error:

```
ERROR: WLC is still running or was shut down abnormally.
```

You can override this error and force the command to execute by adding the `-force` option to the Bulk Loader command file. You can update the existing Bulk Loader command file, or you can copy and update the file, as described in the following procedure:

1. Make a copy, with a new name, of the Bulk Loader command file appropriate for your platform, as shown in the following examples:
 - On Windows:

```
cd BEA_HOME\wlintegration2.1\bin
copy bulkloader.cmd bulkloaderforce.cmd
```
 - On UNIX:

```
cd BEA_HOME/wlintegration2.1/bin
cp bulkloader.sh bulkloaderforce.sh
```
2. Open the new file (in this example, the `bulkloaderforce` command file) in your preferred text editor.

3. Locate the following line in the new file:

```
java %DB_JVMARGS% -classpath %WLICP% com.bea.b2b.bulkloader.BulkLoader -v %CTLFILE%
```

4. Add the `-force` option to this line, as follows:

```
java ... com.bea.b2b.bulkloader.BulkLoader -v -force %CTLFILE%
```

5. Save and close the file.
6. Execute the newly created command file to force the import or export of the desired data.

8 Configuring Persistence and Recovery

The following sections describe how WebLogic Integration provides persistence and recovery for B2B integration applications:

- Understanding Persistence and Recovery
- Configuring B2B Engine Startup
- Configuring Persistence and Recovery

Understanding Persistence and Recovery

The B2B engine must be configured to run in one of the following modes:

- *Persistent mode*

In this mode, persistence is set to `ON`. The in-memory, dynamic state of B2B objects is saved to, and can be retrieved from, persistent storage in the database repository. Run-time states can be recovered in the event of an abnormal shutdown or crash.

- *Nonpersistent mode*

In this mode, persistence is set to `OFF`. The in-memory, dynamic state of B2B objects is not saved. State information does not survive system restarts.

The persistence setting used by the B2B engine is determined by the configuration of the B2B engine startup class. (For instructions on configuring this class, see “Configuring B2B Engine Startup” on page 8-6.) Thus, as soon as you start WebLogic Integration, as described in “Getting Started” in *Starting, Stopping, and Customizing BEA WebLogic Integration*, the B2B engine starts running in either persistent or nonpersistent mode. The setting for persistence cannot be changed as long as the B2B engine is running.

Fortunately, it is not necessary to shut down your entire WebLogic Integration domain in order change the setting for persistence. Only the B2B engine needs to be shut down, and the B2B Console supports shutdown of this single component. (See “Starting and Stopping the B2B Engine from the B2B Console” in *Starting, Stopping, and Customizing BEA WebLogic Integration*). The ability to shut down only the B2B engine is advantageous because it allows you to stop only one functional area of WebLogic Integration; even when the B2B engine is shut down, all other WebLogic Integration applications and resources continue running. When you restart the B2B engine from the B2B console, you are prompted to specify a setting for persistence.

The following sections provide the information you need to understand the consequences of starting or restarting in persistent or nonpersistent mode.

Understanding Persistent Mode

In persistent mode, all state records are read from and written to persistent storage and are not cached in memory. In persistent mode, the B2B engine stores the states of the following types of run-time objects in persistent storage:

- System
- Delivery channels
- Trading partners sessions
- Conversations
- Messages
- Message queues
- Workflow instance
- Tables related to the preceding objects

Like B2B configuration elements, the data required to support persistence is stored in the database repository. The B2B engine creates a state record for each run-time component that it creates during processing. A state record is a row in a database table; the row represents an object state.

The B2B engine caches all persistent state objects in memory, and saves all information necessary to recover these objects in the database. Updates to state objects are mirrored in the database.

Multiple objects can change state as a message is processed by the B2B engine. The B2B engine updates persistent storage for these objects as a group. If a message passes through the B2B engine successfully, then the recorded changes are retained. If message processing fails at any point, the B2B engine can retry some operations. When an operation fails, the B2B engine discards the changes that the operation caused. To maintain and update a set of objects states as a group, the B2B engine uses transactions.

The B2B engine performs special processing for messages. The message size and the large message support configuration determine how to save the message, as follows:

- If the message size exceeds the large message size threshold, and if large message support is enabled, the B2B engine does the following:
 - Saves the message in persistent storage on the file system, rather than in the database
 - Saves the message location in persistent storage in the database
- Otherwise, the B2B engine saves the entire message in persistent storage.

For information about the large message threshold and large message support, see [“Configuring B2B Integration”](#) in *Online Help for the WebLogic Integration B2B Console*.

Understanding Nonpersistent Mode

In nonpersistent mode, all state records other than those in the repository are maintained in memory and recovery is disabled. In nonpersistent mode, the B2B engine uses only in-memory tables to store the states of run-time objects. The only difference between persistent mode and nonpersistent mode is that in nonpersistent mode the B2B engine does not save states in persistent storage. This means that states cannot be retrieved when the system is shut down. As a result, state information does not survive system restarts.

Notes: We recommend that you use nonpersistent mode only when developing your WebLogic Integration B2B application, not in production.

If you migrate a Java messaging application written with the WebLogic Collaborate C-Enabler API, the migrated application must be run in a separate JVM in nonpersistent mode.

Understanding Recovery

When you restart after the B2B engine has been shut down, the recovery behavior is dependent on:

- Whether the B2B engine was running in persistent or nonpersistent mode before it was shut down
- Whether the shutdown was normal or abnormal

Note: When a crash occurs, you must resolve the cause of the crash and restart B2B integration to recover and reinitialize the persistent data in the run-time environment.

- Whether you are restarting in persistent or nonpersistent mode

You can shut down the B2B engine in either of two ways: by shutting down WebLogic Integration entirely, using the `stopWeblogic` command, or by shutting down the only B2B engine from the B2B Console. When the B2B engine is successfully shut down using either of these methods, the shutdown is considered normal. For additional information about the shutdown options, see [“Stopping WebLogic Integration”](#) and [“Starting and Stopping the B2B Engine from the B2B Console”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

In a normal shutdown, the state tables are emptied.

When the B2B engine is restarted after a shutdown, its behavior depends on whether it is restarted in persistent mode or nonpersistent mode:

- If the B2B engine starts in nonpersistent mode, it does not undertake any particular recovery activities.
- If the B2B engine starts in persistent mode, it tries to read state records from persistent storage to determine whether or not to perform the recovery procedure. If the B2B engine shutdown was normal, or if it was not running in persistent mode at the time of the shutdown, no state records are found. Otherwise, the B2B engine performs the following recovery procedure:
 - a. Reinstantiates active run-time objects, such as delivery channels and trading partner sessions.
 - b. Terminates conversations that expired while the B2B engine was down.
 - c. Restarts message queues for active trading partners.

Configuring B2B Engine Startup

The configuration of the B2B engine startup class determines the persistence setting the B2B engine uses when you start WebLogic Integration. The following procedure describes how to view or modify the B2B engine startup configuration from the WebLogic Server Administration Console.

To view or modify the startup configuration:

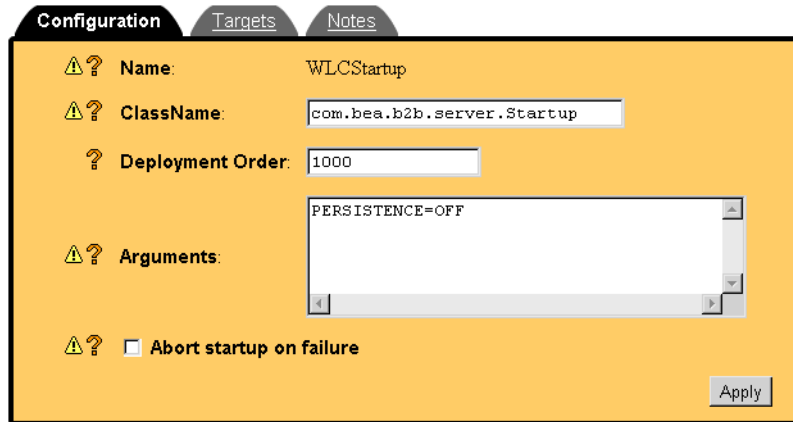
1. Start the WebLogic Server Administration Console, as described in [“Starting the WebLogic Server Administration Console”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.
2. In the navigation tree, choose Deployments→Startup & Shutdown →WLCStartUp.
3. Select the Configuration tab.

The WLCStartUp configuration is displayed. One of the following is displayed in the Arguments text box:

- PERSISTENCE=ON
The B2B engine is configured to start in persistent mode.
- PERSISTENCE=OFF
The B2B engine is configured to start in nonpersistent mode.

For example, in the configuration shown in the following figure, the B2B engine is configured to start in nonpersistent mode.

Figure 8-1 Startup Configuration



4. To modify the current setting, edit as required.
5. Close the WebLogic Server Administration Console.
6. To initiate the new settings, shut down and restart WebLogic Integration as follows:
 - Follow the shutdown instructions in [“Stopping WebLogic Integration”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.
 - Follow the restart instructions in [“Starting WebLogic Integration”](#) in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

You can also update the startup class by editing the `config.xml` file. If you use this method, you must shut down WebLogic Integration before you edit the file. Any changes made to the file while WebLogic Integration is running will be lost.

To modify the startup class in the `config.xml` file:

1. Make sure WebLogic Integration has been shut down.
2. Open the `config.xml` file for the domain in your preferred text editor.
3. Locate the following entry:

```
<StartupClass
  ClassName="com.bea.b2b.server.Startup"
  Name="WLCStartup"
  Arguments="PERSISTENCE=OFF"
  Targets="myserver" />
```

4. Edit to set persistence on or off, as required.

Configuring Persistence and Recovery

The following table summarizes how to set the persistence mode when you start the B2B engine. For details about the start procedures referenced in the table, see “Starting WebLogic Integration” and “Starting and Stopping the B2B Engine from the B2B Console” in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Table 8-1 Setting the Persistence Mode

When you start the B2B engine from the . . .	Do the following to start in persistent mode . . .	Or do the following to start in nonpersistent mode . . .
B2B Console	<ol style="list-style-type: none"> 1. Select the high-level Monitoring tab. 2. Select the nested General tab. 3. Click <code>Start this server</code>. A dialog box is displayed. 4. In the dialog box, make sure the Persistence On option is selected. 5. Click Yes. 	<ol style="list-style-type: none"> 1. Select the high-level Monitoring tab. 2. Select the nested General tab. 3. Click <code>Start this server</code>. A dialog box is displayed. 4. In the dialog box, make sure the Persistence On option is selected. 5. Click Yes.
Start menu or command line	<p>Start WebLogic Integration in one of the following ways:</p> <ul style="list-style-type: none"> ■ Run the <code>startWebLogic</code> command. ■ Use the Start Server shortcut. <p>Note: By default, Persistence is on in the WebLogic Integration preconfigured domains.</p>	<ol style="list-style-type: none"> 1. Set Persistence to <code>OFF</code>, as described in “Configuring B2B Engine Startup” on page 8-6. 2. Start WebLogic Integration in one of the following ways: <ul style="list-style-type: none"> ■ Run the <code>startWebLogic</code> command. ■ Use the Start Server shortcut.

A Update Considerations

Table A-1 summarizes details about configuration parameters that you should keep in mind when updating your configuration of the B2B engine, trading partners, conversation definitions, collaboration agreements, logic plug-ins, and business protocols.

Table A-1 B2B Configuration Update Considerations

Element	Category	Parameter	Update Considerations
B2B	General	B2B engine name	Cannot be updated by the user.
		Description	Can be updated at any time, but changes do not become effective until WebLogic Server is shut down and restarted.
		Large Message Support	
	Security	System Password	Can be updated at any time, but changes do not become effective until WebLogic Server is shut down and restarted.
		Audit Log Class	
		Certificate Verification Class	
		Secure Timestamp Class	
		Certificate Authority Directory	

A *Update Considerations*

Table A-1 B2B Configuration Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
B2B (Continued)	Proxy	Host	Can be updated at any time, but changes do not become effective until WebLogic Server is shut down and restarted.
		Port	
	Preferences	Number of items displayed per page (1-50)	Can be updated at any time. The update becomes effective immediately.
		Default retry value	
		Default retry interval	
		Default timeout value	
		Hide advanced configuration controls	
Display entities in the navigation tree			

Table A-1 B2B Configuration Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Trading Partners	General	Name	Cannot be updated. Workaround: Delete the obsolete trading partner name and create a new name in its place. Note: You cannot delete a trading partner if it is engaged in a trading partner session.
		Description	Can be updated at any time. The update becomes effective immediately.
		Type	
		Address	
		Email	
		Phone	
		Fax	
		WLS User Name	Can be updated at any time, but the update does not become effective until WebLogic Server is shut down and restarted.
		State	Can be updated at any time. The update becomes effective immediately.
		Party IDs	Party ID
Business ID	Cannot be updated if the trading partner is engaged in a trading partner session.		
Business ID Type			
Certificates	Certificate Name	Certificate Name	A certificate can be added at any time, but the addition is not available until WebLogic Server is shut down and restarted.
		Certificate Type	
		Certificate Location	An existing certificate cannot be updated if it is in the security cache.
		Private Key Location	

Table A-1 B2B Configuration Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Trading Partners (Continued)	Doc Exchange	Document Exchange Name	A document exchange can be added at any time.
		Business Protocol Binding	An existing document exchange cannot be updated while the B2B engine is running.
		Business Protocol Definition	
		End Point Type	
	Doc Exchange XOCP Specific Settings	Confirmed Delivery	Cannot be updated while the B2B engine is running.
		Message History	
		Retries	
	Doc Exchange cXML Specific Settings	Digital Signature (Nonrepudiation)	A digital certificate can be added at any time, but the addition is not available until WebLogic Server is shut down and restarted. An existing certificate cannot be updated if it is in the security cache.
		Shared Secret	Cannot be updated while the B2B engine is running.
	Doc Exchange RosettaNet 1.1 Specific Settings	Signature Certificate	
Digital Signature (Nonrepudiation)		A digital signature can be added at any time, but the new signature is not available until WebLogic Server is shut down and restarted. An existing certificate cannot be updated if it is in the security cache.	
Doc Exchange RosettaNet 2.0 Specific Settings	Encryption	An encryption certificate can be added at any time, but the addition is not available until WebLogic Server is shut down and restarted. An existing certificate cannot be updated if it is in the security cache.	
	Digital Signature (Nonrepudiation)	A digital certificate can be added at any time, but the addition is not available until WebLogic Server is shut down and restarted. An existing certificate cannot be updated if it is in the security cache.	

Table A-1 B2B Configuration Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Trading Partners (Continued)	Transport	Transport Name	A transport can be added at any time.
		Transport Protocol	Cannot be updated while the B2B engine is running.
		Security Protocol	
		Endpoints	
	Delivery Channels	Delivery Channel Name	Cannot be updated while the B2B engine is running.
		Transport	
		Document Exchange	
	Advanced XOCP Filters & Routers	XOCP Filter XPath Expressions	Can be updated at any time. The update becomes effective immediately.
		XOCP Router XPath Expressions	
	Advanced Extended Properties	Property Name	Can be updated at any time. The update becomes effective immediately.
		Property Value	
		Attributes	
Conversations	General	Name	Cannot be updated. Workaround: Delete the obsolete conversation definition name and create a new name in its place. Note: You cannot delete a conversation definition if an active conversation is using the definition.
		Version	Conversation definition parameters cannot be updated if there is an active conversation using the conversation definition.
		Description	
		Business Protocol	
		Default Timeout	
		Roles	

A Update Considerations

Table A-1 B2B Configuration Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Collaboration Agreements	General	Collaboration Agreement Name	Cannot be updated. Workaround: Delete the obsolete agreement name and create a new name in its place. Note: You cannot delete a collaboration agreement if it is registered.
		Description	Can be updated at any time. The update becomes effective immediately.
		Version	Cannot be updated when the collaboration agreement is registered.
	Parties	Conversation Definition	
		Trading Partner	An existing party definition cannot be updated while the B2B engine is running.
		Role	Parties can be added at any time.
Logic Plug-Ins	General	Name	Cannot be updated. Workaround: Delete the obsolete logic plug-in name and create a new name in its place. Note: When you delete a logic plug-in, the plug-in remains in use until WebLogic Server is shut down and restarted. Custom logic plug-in definitions can be added at any time.
		Description	Can be updated at any time, but the updates do not become effective until WebLogic Server is shut down and restarted.
		Type	
	Java Class Name		
	Parameters for Java Class		

Table A-1 B2B Configuration Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Business Protocol	General	Name	Cannot be updated by the user.
		Description	
		Business Protocol	
	Filters & Routers	Java Class Name	Custom logic plug-ins can be added to the router or filter chain, and existing plug-ins can be updated at any time, but the updates do not become effective until WebLogic Server is shut down and restarted.
		Filter Chain	
		Router Chain	
	XOCP Filters & Routers	XOCP Filter XPath Expressions	XPath filter and router expressions can be added to the XOCP business protocol definition, but they do not become effective until WebLogic Server is shut down and restarted.
		XOCP Router XPath Expressions	
