BEA WebLogic Collaborate

A Component of BEA WebLogic Integration

Administering BEA WebLogic Collaborate

BEA WebLogic Collaborate Release 2.0
Document Edition 2.0
July 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Administering BEA WebLogic Collaborate

Document Edition	Date	Software Version
2.0	July 2001	2.0

Contents

2. Configuration Requirements

3. Basic Configuration Tasks

4. Advanced Configuration Tasks

5. Importing and Exporting Business Collaborations

6. Monitoring WebLogic Collaborate

7. Working with the Repository

8. Working with the Bulk Loader

About This Document

This document describes how to configure and manage WebLogic Collaborate™, the Business-to-Business (B2B) component of WebLogic Integration.

Specifically, it discusses following topics:

- Chapter 1, “Starting, Stopping, and Customizing WebLogic Collaborate,” provides the basic information you need to get started with WebLogic Collaborate.
- Chapter 2, “Configuration Requirements,” presents sample configurations that summarize the requirements for typical collaboration participants.
- Chapter 3, “Basic Configuration Tasks,” provides an overview of the tasks and procedures required to configure WebLogic Collaborate for trading exchange, supply chain management, and collaborative commerce applications.
- Chapter 4, “Advanced Configuration Tasks,” serves as an introduction to the more advanced features of WebLogic Collaborate and provides a roadmap to where more detailed information can be found.
- Chapter 5, “Importing and Exporting Business Collaborations,” describes how you can export and import the necessary components of a business collaboration.
- Chapter 6, “Monitoring WebLogic Collaborate,” provides an overview of how you can use the WebLogic Collaborate Administration Console to monitor and control WebLogic Collaborate, trading partner sessions, delivery channels, conversations, and collaboration agreements.
- Chapter 7, “Working with the Repository,” describes how WebLogic Collaborate configuration elements are stored in the database repository.
- Chapter 8, “Working with the Bulk Loader,” provides the information you need to use the bulkloader to load data into the repository.

- Chapter 9, “Configuring Persistence and Recovery,” describes how persistence and recovery work in WebLogic Collaborate.
- Chapter A, “WebLogic Collaborate Sample Configuration Files,” provides system administrators with samples of the files that configure WebLogic Server and deploy the applications required for WebLogic Collaborate.
- Chapter B, “Update Considerations,”summarizes the constraints and considerations related to modifying WebLogic Collaborate 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 Collaborate.

Before reading this document, we recommend that you read the *Introducing BEA WebLogic Collaborate* document.

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 Collaborate 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 Collaborate 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/.

Related Information

The WebLogic Collaborate document set includes the following:

- *BEA WebLogic Collaborate Release Notes*
- *Introducing BEA WebLogic Collaborate*
- *Administering BEA WebLogic Collaborate*
- *Creating Workflows for BEA WebLogic Collaborate*
- *Using BEA WebLogic Collaborate Security*
- *Using BEA WebLogic Collaborate Samples*
- *Writing Messages to the BEA WebLogic Collaborate Log*
- *Implementing cXML for BEA WebLogic Collaborate*
- *Implementing RosettaNet for BEA WebLogic Collaborate*
- *Migrating BEA WebLogic Collaborate to Release 2.0*

- *Programming BEA WebLogic Collaborate Messaging Applications*
- *Programming BEA WebLogic Collaborate Logic Plug-Ins*
- *Programming BEA WebLogic Collaborate Management Applications*
- *BEA WebLogic Collaborate Administration Console Online Help*
- *BEA WebLogic Collaborate Glossary*

Contact Us!

Your feedback on the BEA WebLogic Collaborate 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 Collaborate documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Collaborate 2.0 release.

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

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

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

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

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR

Convention	Item
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Starting, Stopping, and Customizing WebLogic Collaborate

This section provides the information you need to start, stop, and customize WebLogic Collaborate. It includes the following topics:

- Understanding the Preconfigured Domains
- Understanding WebLogic Collaborate Users and Passwords
- Starting WebLogic Collaborate
- Starting the WebLogic Collaborate Administration Console
- Stopping WebLogic Collaborate
- Starting the WebLogic Server Administration Console
- Starting WebLogic Process Integrator Studio
- Customizing WebLogic Collaborate
- Verifying Configuration

Understanding the Preconfigured Domains

A WebLogic Server domain is a collection of WebLogic Server resources and applications that occupy a single namespace. The configuration of a domain and the applications deployed are dependent on the domain start and configuration files. Components of WebLogic Integration (such as WebLogic Collaborate) are applications that are deployed in a WebLogic Server domain.

When you install WebLogic Collaborate, the following WebLogic Collaborate sample and custom domains are created and configured:

- *WebLogic Collaborate Sample Domain*
  This domain is located in the `/config/samples` directory under the WebLogic Collaborate installation directory. For example, if you installed the product in the default location, the sample domain would be located in the `<BEA_Home>/wlintegration2.0/collaborate/config/samples` directory.
- *WebLogic Collaborate Custom Domain*
  This domain is located in the `/config/mydomain` directory under the WebLogic Collaborate installation directory. For example, if you installed the product in the default location, the custom domain would be located in `<BEA_Home>/wlintegration2.0/collaborate/config/mydomain` directory.

The sample domain is configured with a JDBC connection pool that provides the access information required to connect to a Cloudscape database. (Cloudscape is a pure-Java relational database management system that BEA ships with the WebLogic distribution to allow you to run code examples.) The installation program creates a Cloudscape database for this domain, and populates the database with the tables WebLogic Collaborate requires. No further configuration of this domain is required to run the sample applications included with the WebLogic Collaborate distribution or obtained from the WebLogic Developer Center.

The custom domain is configured with a JDBC connection pool that provides the access information required to connect to the database selected during installation. If you selected a database other than Cloudscape for the custom domain, you must execute the appropriate scripts as described in *Installing BEA WebLogic Collaborate* to create the database tables WebLogic Collaborate requires. This custom domain can then be used to run sample applications or to develop and deploy your own applications.

A copy of each of the following files is located in the sample and custom domain directories. These files control the WebLogic Server resource configuration and the initial deployment of the applications in each domain:

- Start command—each domain contains a `startWeblogic.cmd` (Windows) or `startWeblogic.sh` (UNIX) file.
- Environment command—the WebLogic Collaborate environment is set by the `setEnv.cmd` (Windows) or `setEnv.sh` (UNIX) file. The start command calls this file.
- `config.xml` file—an XML document that describes the configuration of an entire WebLogic Server domain.
- The `fileRealm.properties` file—the default security realm in WebLogic Server is the File realm. The User, Group, and ACL objects that are created when WebLogic Server is started are stored in the `fileRealm.properties` file.

An example of each file is provided in Appendix A, "WebLogic Collaborate Sample Configuration Files."

As part of the WebLogic Collaborate installation, each `setEnv` file is updated with the information required to start WebLogic Collaborate. For information about the required environment settings, see "Updating the WebLogic Collaborate Environment" on page 1-30.

When you start WebLogic Collaborate by executing the `startWebLogic` command for one of the preconfigured domains (as described in "Starting WebLogic Collaborate" on page 1-5), WebLogic Collaborate, WebLogic Process Integrator, and the associated WebLogic Collaborate plug-in, are deployed on WebLogic Server.

As you become more familiar with WebLogic Collaborate and the other WebLogic Integration components, you can use the custom or samples domain as the starting point for development and testing, or you can create your own domain by modeling it after one of the preconfigured domains.

Understanding WebLogic Collaborate Users and Passwords

The WebLogic Collaborate installation configures a default fileRealm security realm for each of the domains described in the previous section. The following default users and passwords are configured.

Table 1-1 Default User Names and Passwords

User Name	Default Password	Description	Update
system	security	This login is used to: ■ Boot the WebLogic Server. ■ Access the WebLogic Server Administration Console. ■ Access the WebLogic Collaborate Administration Console.	Update this password from the WebLogic Server Administration Console.
wlcsystem	wlcsystem	This login is used by the WebLogic Collaborate run-time environment.	Update this password from the WebLogic Collaborate Administration Console. **Warning:** Do not update this password using the WebLogic Server Administration Console.
admin	security	These default logons are used to log on and connect to WebLogic Process Integrator from the Studio or Worklist client.	Update users (add or delete) from the WebLogic Process Integrator Studio. Update an existing user password from the WebLogic Server Administration Console.
mary	password		
joe	password		

To ensure system security, you can change the default passwords as described in "Updating the Default Passwords" on page 1-31.

Starting WebLogic Collaborate

As described in "Understanding the Preconfigured Domains" on page 1-2, when you execute the `startWeblogic.cmd` (Windows) or `startWeblogic.sh` (UNIX) command for a domain, WebLogic Server is started and deploys WebLogic Collaborate, WebLogic Process Integrator, and the WebLogic Collaborate plug-in for WebLogic Process Integrator.

The following sections provide instructions for executing the start command on a Windows or UNIX system.

Note: If you have shut down the WebLogic Collaborate application from the WebLogic Collaborate Administration Console, this procedure does not apply. See "Restarting from the WebLogic Collaborate Administration Console" on page 1-15.

Starting WebLogic Collaborate on Windows

On a Windows system, you can start WebLogic Collaborate with the program icons or from the command line.

Note: If WebLogic Server is already running, you must stop it before executing the start command.

Starting WebLogic Collaborate with Program Icons

To start WebLogic Collaborate using program icons, do one of the following:

- To start from the WebLogic Collaborate custom domain, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Collaborate→Start Collaborate.
- To start from the WebLogic Collaborate samples domain, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Collaborate→Samples→Start Collaborate.

A command window is launched, and startup messages are displayed. When the following message is displayed, startup has completed successfully:

```
<Month DD, YYYY hh:mm:ss meridianAbbreviation Timezone> <Notice>
<WebLogicServer> <ListenThread listening on port 7001>
```

Starting WebLogic Collaborate from the Command Line

To start WebLogic Collaborate from the command line:

1. Choose Start→Run.
2. Enter `cmd` in the Open text box then click OK.
3. Go to the appropriate domain subdirectory.

   For example, if you installed the product in the default location, and you want to start the WebLogic Collaborate custom domain, enter the following:

   ```
   cd \bea\wlintegration2.0\collaborate\config\mydomain
   ```

   For the default location for each preconfigured domain see "Understanding the Preconfigured Domains" on page 1-2.
4. Execute the `startWeblogic` command by entering:

   ```
   startWeblogic
   ```

   When the following message is displayed, startup has completed successfully:

```
<Month DD, YYYY hh:mm:ss meridianAbbreviation Timezone> <Notice>
<WebLogicServer> <ListenThread listening on port 7001>
```

Starting WebLogic Collaborate on UNIX

Note: If WebLogic Server is already running, you must stop it before executing the start command.

To start WebLogic Collaborate on a UNIX system:

1. Go the appropriate domain subdirectory.

   For example, if you installed the product in the default location, and you want to start the WebLogic Collaborate custom domain, enter the following:

   ```
   cd <BEA_Home>/wlintegration2.0/collaborate/config/mydomain
   ```

   For the default location for each preconfigured domain see "Understanding the Preconfigured Domains" on page 1-2.

2. Execute the `startWeblogic.sh` command by entering:

   ```
   . ./startWeblogic.sh
   ```

   When the following message is displayed, startup has completed successfully:

```
<Month DD, YYYY hh:mm:ss meridianAbbreviation Timezone> <Notice>
<WebLogicServer> <ListenThread listening on port 7001>
```

Starting the WebLogic Collaborate Administration Console

You can use the WebLogic Collaborate Administration Console to:

- Configure WebLogic Collaborate preferences, trading partners, conversation definitions, collaboration agreements, business protocol definitions, and logic plug-ins
- Export and import configured elements
- Monitor and control WebLogic Collaborate, trading partner sessions, conversations, and collaboration agreements

Note: The WebLogic Collaborate Administration Console requires Netscape Navigator 4.7 (or above) or Microsoft Internet Explorer 5.x (or above). The browser you use must be configured to accept cookies. In Netscape Navigator select File→Preferences to display the Preferences dialog box. The accept cookies option is set in the Advanced category. In Microsoft Internet Explorer, select Tools→Internet Options to display the Internet Options dialog box. Cookies are enabled by customizing the security level settings for the applicable Web content zone.

To start the WebLogic Collaborate Administration Console:

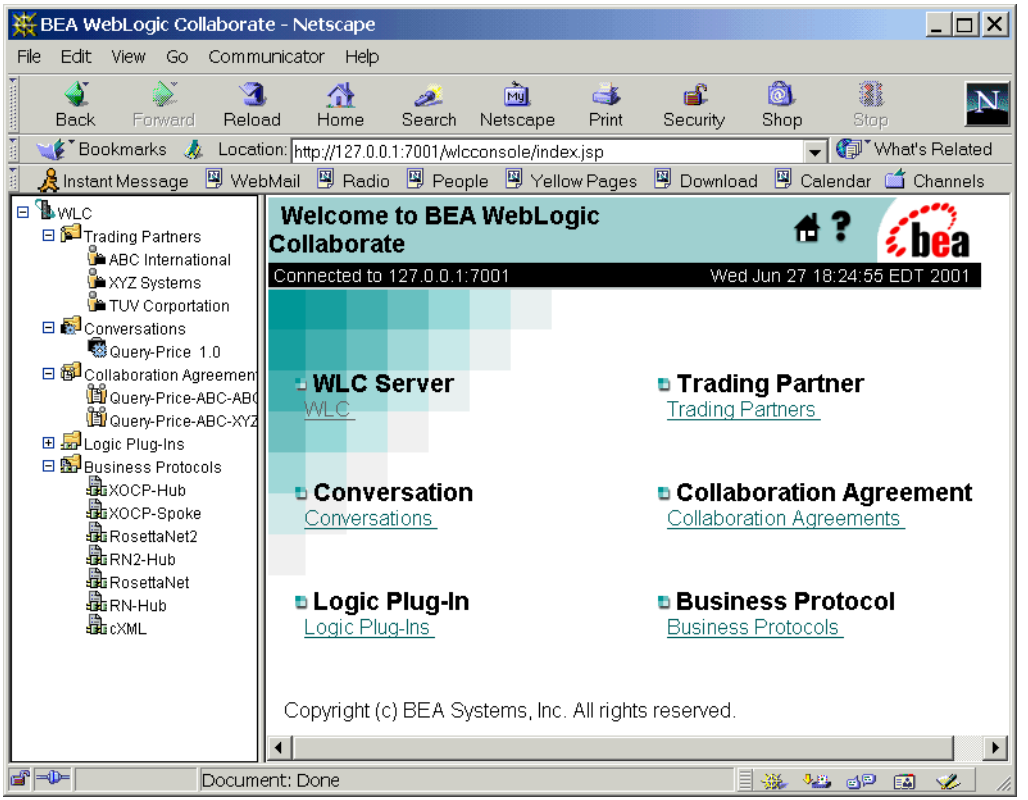
1. Start WebLogic Collaborate as described in "Starting WebLogic Collaborate" on page 1-5.
2. Do one of the following:
   - On a Windows or UNIX system, open the following URL in your Web browser:

     `http://host:7001/wlcconsole`

     In this URL, *host* is the computer name or IP address of the system that is running WebLogic Collaborate, 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.
   - On a Windows system, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Collaborate→Start Admin Console.
3. When prompted, enter `system` in the user name field, then enter the `system` password.

   Note: If you have not yet changed the password, see "Understanding WebLogic Collaborate Users and Passwords" on page 1-4 for the default password.
4. The WebLogic Collaborate Administration Console is displayed in your Web browser as shown in the following figure.

Figure 1-1 WebLogic Collaborate Administration Console



Note: Bookmark the WebLogic Collaborate Administration Console for quick access.

Overviews of the tasks performed from the WebLogic Collaborate Administration Console are provided in Chapter 3, “Basic Configuration Tasks,” Chapter 4, “Advanced Configuration Tasks,” and Chapter 6, “Monitoring WebLogic Collaborate.”

Detailed information about using the interface and setting the required attributes is provided in the WebLogic Collaborate Administration Console online help. For help at any time, click the question mark in the upper right-hand corner of the page.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

Stopping WebLogic Collaborate

There are two options for stopping WebLogic Collaborate:

- Shut down WebLogic Server and all deployed applications by executing the `stopWeblogic.cmd` (Windows) or `stopWebLogic.sh` (UNIX) command.
- Shut down only the WebLogic Collaborate application from the WebLogic Administration Console.

The required procedures are provided in the following sections:

- Stopping WebLogic Server and All Deployed Applications
- Stopping from the WebLogic Collaborate Administration Console
- Restarting from the WebLogic Collaborate Administration Console

Stopping WebLogic Server and All Deployed Applications

You can stop WebLogic Server and all applications deployed in a WebLogic Collaborate domain by executing the `stopWeblogic.cmd` (Windows) or `stopWeblogic.sh` (UNIX) command located in the domain directory. The following sections provide instructions for executing the stop command on a Windows or UNIX system.

Stopping WebLogic Server and All Deployed Applications on Windows

The following sections provide instructions for stopping WebLogic Server and all deployed applications by using the program icons or by executing the `stopWeblogic.cmd` from the command line.

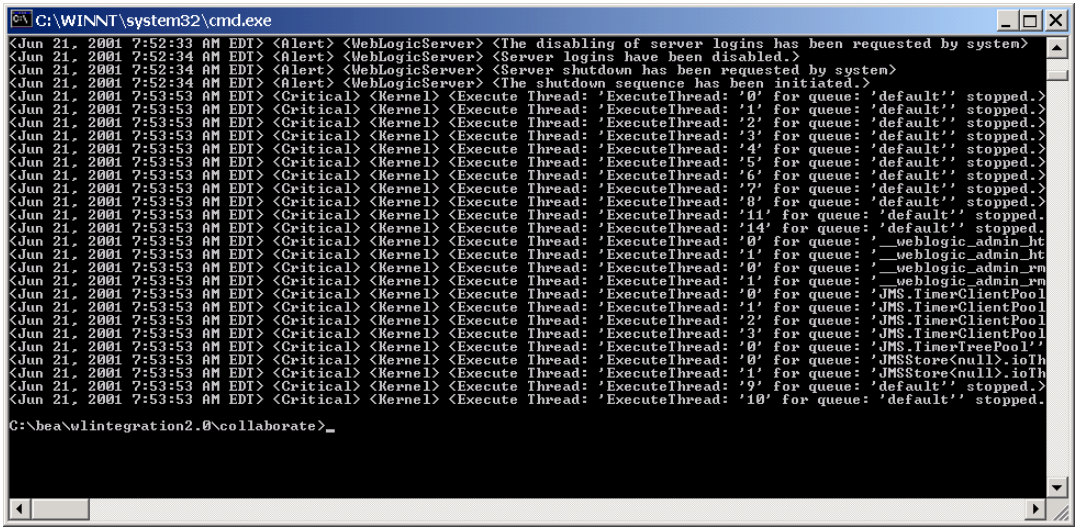
Stopping WebLogic Server and All Deployed Applications with Program Icons

To stop WebLogic Server and all deployed applications using program icons, do one of the following:

- To stop the custom domain, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Collaborate→Stop Collaborate.
- To stop the samples domain, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Collaborate→Samples→Stop Collaborate.

Shutdown messages are displayed as shown in the following figure. When you are returned to the command prompt, shut down is complete.

Figure 1-2 Shut down Sequence Messages



Once WebLogic Server and all applications have been terminated, you can restart WebLogic Collaborate, as described in "Starting WebLogic Collaborate on Windows" on page 1-5.

Stopping WebLogic Server and All Deployed Applications from the Command Line

To start WebLogic Collaborate from the command line:

1. Choose Start→Run
2. Enter `cmd` in the Open text box then click OK.
3. Go to the appropriate domain subdirectory.

   For example, if you installed the product in the default location, and you want to start the WebLogic Collaborate custom domain, enter the following:

   ```
   cd \bea\wlintegration2.0\collaborate\config\mydomain
   ```

   For the default location for each preconfigured domain see "Understanding the Preconfigured Domains" on page 1-2.
4. Execute the `stopWeblogic` command by entering:

   ```
   stopWeblogic
   ```

   Shutdown messages are displayed as shown in Figure 1-2.

Once WebLogic Server and all applications have been terminated, you can restart WebLogic Collaborate, as described in "Starting WebLogic Collaborate on Windows" on page 1-5

Stopping WebLogic Server and All Deployed Applications on UNIX

To stop WebLogic Server and all deployed applications on a UNIX system:

1. Go the appropriate domain subdirectory.

   For example, if you installed the product in the default location, and you want to stop the WebLogic Collaborate custom domain, enter the following:

   ```
   cd <BEA_Home>/wlintegration2.0/collaborate/config/mydomain
   ```

   For the default location for each preconfigured domain see "Understanding the Preconfigured Domains" on page 1-2.
2. Execute the `stopWeblogic.sh` command by entering:

   ```
   . ./stopWeblogic.sh
   ```

   Shutdown messages are displayed as shown in Figure 1-2.

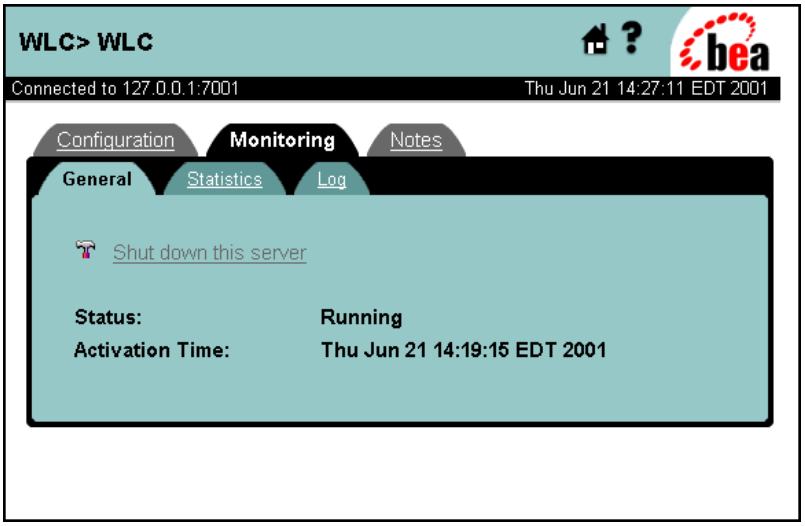
Stopping from the WebLogic Collaborate Administration Console

You can shut down the WebLogic Collaborate application, but leave WebLogic Server and other deployed applications running, from the WebLogic Collaborate Administration Console.

To shut down the WebLogic Collaborate application:

1. Select WebLogic Collaborate (WLC) from the navigation tree to display the WLC page.
2. Select the Monitoring tab, then select the General tab to display the Monitoring General tab as shown in the following figure.

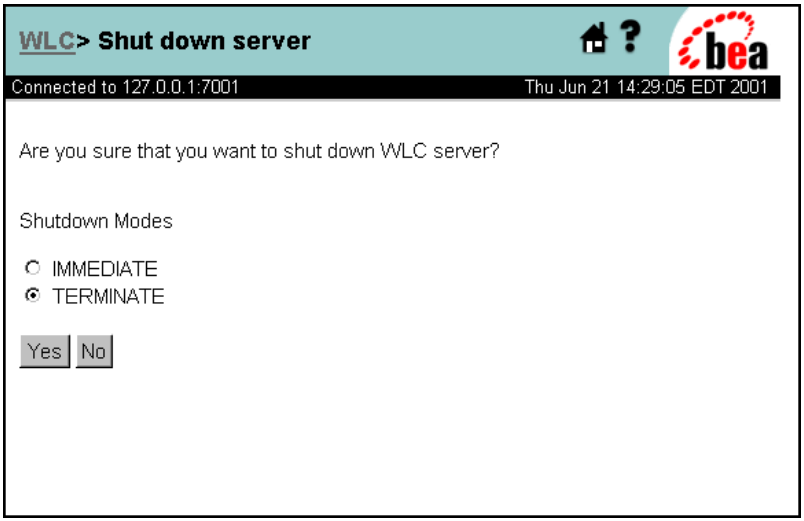
Figure 1-3 WebLogic Collaborate Monitoring General Tab



3. Select Shut down this server.

   The shut down confirmation page is displayed. This page allows you to select shut down mode as shown in the following figure.

Figure 1-4 Shut down Options



4. Select the appropriate shut down mode.

   - *Terminate*
     This mode shuts down active delivery channels which triggers the termination of the associated trading partner sessions. The termination of the trading partner sessions triggers termination of the associated conversations and removal of queues.
   - *Immediate*
     This mode stops all activity immediately.

5. Click Yes to shut down WebLogic Collaborate.

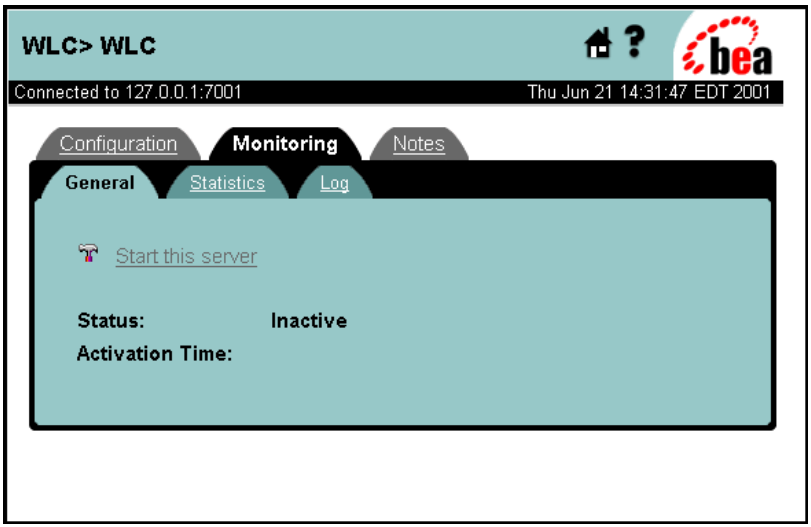
   The WebLogic Collaborate application is shut down. WebLogic Server and any other applications remain active. When WebLogic Collaborate has shut down, you can restart it as described in the following procedure.

Restarting from the WebLogic Collaborate Administration Console

To restart WebLogic Collaborate after shutting down from the WebLogic Collaborate Administration Console:

1. Select WebLogic Collaborate (WLC) from the navigation tree.
2. Select the Monitoring tab, then select the General tab to display the Monitoring General tab as shown in the following figure.

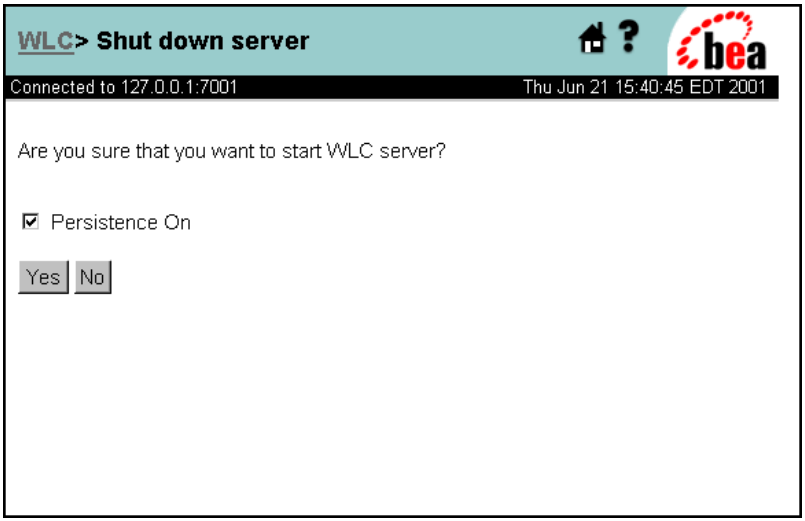
Figure 1-5 WebLogic Collaborate Monitoring General Tab



3. Select Start this server.

   The start up confirmation page is displayed. This page allows you to turn persistence on or off as shown in the following figure.

Figure 1-6 Start Up Confirmation



4. Do one of the following:
   - Click Yes to start with persistence on.
   - Deselect Persistence On, then Click Yes to start with persistence off.

   For information about WebLogic Collaborate persistence, see Chapter 9, "Configuring Persistence and Recovery."

The restart process may take a couple minutes. You will be returned to the Monitoring General tab when the process is complete.

Starting the WebLogic Server Administration Console

Some tasks require access to the WebLogic Server Administration Console for the active domain. The following procedure describes how to start the WebLogic Server Administration Console.

To start the WebLogic Server Administration Console:

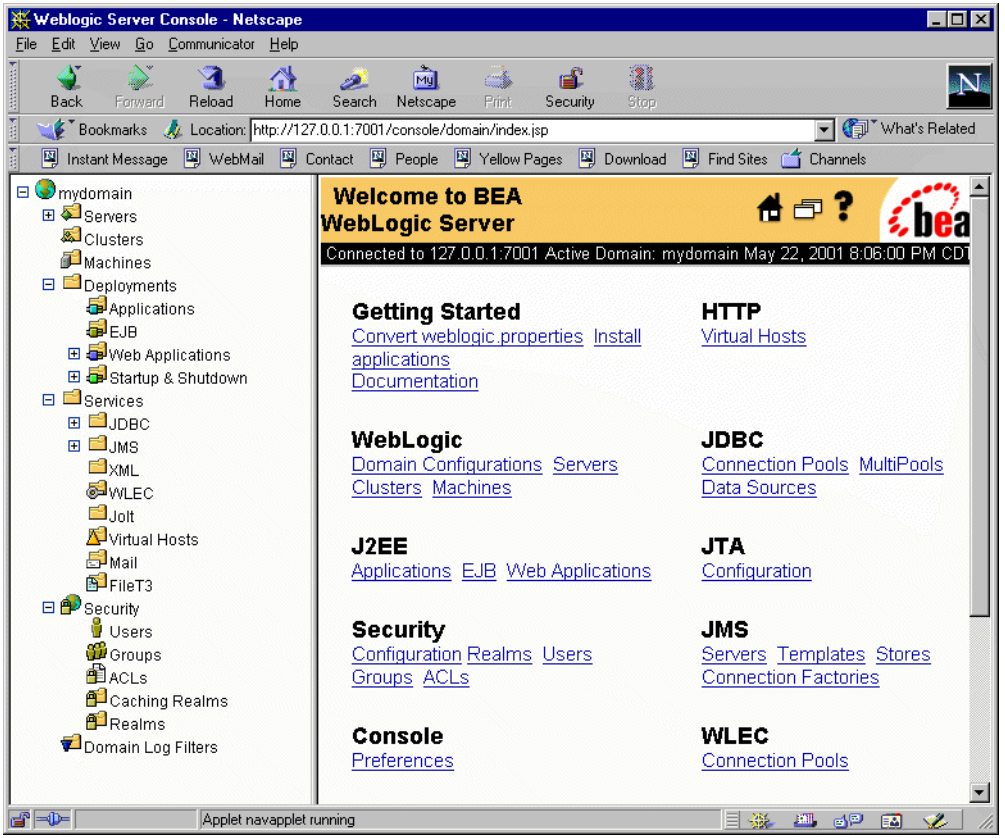
1. Start WebLogic Collaborate as described in "Starting WebLogic Collaborate" on page 1-5.
2. Do one of the following:
   - On a Windows or UNIX system, open the following URL in your Web browser:

     `http://host:7001/console`

     In this URL, *host* is the computer name or IP address of the system that is running WebLogic Collaborate and `7001` is the listen port configured. Specify `localhost` or `127.0.0.1` if WebLogic Collaborate is running on the local computer.
   - On a Windows system, choose Start →Programs →BEA WebLogic E-Business Platform →WebLogic Server 6.0→Start Default Console
3. When prompted, enter `system` in the user name field; then enter the `system` password.

   Note: The password you enter is the WebLogic Server `system` password specific to the active domain. It is usually not the same as the default password specified when you install WebLogic Server. If you have not yet changed the password, see "Understanding WebLogic Collaborate Users and Passwords" on page 1-4 for the default password.
4. The WebLogic Server Administration Console is displayed in your Web browser as shown in the following figure.

Figure 1-7 WebLogic Server Administration Console



Note: Bookmark the WebLogic Server Administration Console for quick access.

For instructions on using and navigating the WebLogic Server Administration Console, see the WebLogic Server 6.0 documentation set.

Starting WebLogic Process Integrator Studio

WebLogic Process Integrator Studio, with the WebLogic Collaborate plug-in, is used to create the collaborative workflows that implement the trading partner roles assigned in WebLogic Collaborate.

Both the custom and sample WebLogic Collaborate domains contain a `studio.cmd` (Windows) or `studio.sh` (UNIX) command file. This command can be used to start WebLogic Process Integrator Studio with the WebLogic Collaborate plug-in if you have installed the Studio client in the same BEA Home directory as the WebLogic Collaborate installation.

If you have installed the Studio client on another system, or will be deploying WebLogic Collaborate in another domain, you need to be aware of certain requirements.

The following sections provide the information you need to start the WebLogic Process Integrator Studio:

- Starting Studio for the Custom or Samples Domain
- Starting Studio on a System Remote to WebLogic Collaborate
- Logging In and Connecting to the WebLogic Process Integrator Server
- Studio Configuration Requirements

Starting Studio for the Custom or Samples Domain

The following sections provide instructions for starting a locally installed Studio client for the custom or sample domain on a Windows or UNIX system. If you have installed Studio on a system that is remote to the WebLogic Collaborate installation, see "Starting Studio on a System Remote to WebLogic Collaborate" on page 1-21.

Starting on Windows

To start the local installation of the WebLogic Process Integrator Studio on Windows:

1. Do one of the following:
   - For the WebLogic Collaborate custom domain, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Collaborate→Start Studio
   - For the WebLogic Collaborate samples domain, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Collaborate→Samples→Start Studio

   The `studio.cmd` executes in a command window, then, within a few moments, the Studio application window is displayed.
2. Log in as described in "Logging In and Connecting to the WebLogic Process Integrator Server" on page 1-22.

Starting on UNIX

To start the local installation of the WebLogic Process Integrator Studio on UNIX:

1. Do one of the following:
   - For the WebLogic Collaborate custom domain, go to the `collaborate/config/mydomain/shortcuts` directory, then run the `studio.sh` command. For example, if you installed the product in the default location you would enter the following:

```
cd <BEA_Home>/wlintegration2.0/collaborate/config/mydomain/shortcuts
    . /studio.sh
```

   - For the WebLogic Collaborate samples domain, go to the `collaborate/config/samples/shortcuts` directory, then run the `studio.sh` command. For example, if you installed the product in the default location you would enter the following:

```
cd <BEA_Home>/wlintegration2.0/collaborate/config/samples/shortcuts
    . /studio.sh
```

   The `studio.sh` file executes, then, within a few moments, the Studio application window is displayed.
2. Log in as described in "Logging In and Connecting to the WebLogic Process Integrator Server" on page 1-22.

Starting Studio on a System Remote to WebLogic Collaborate

Before starting Studio on a system remote to WebLogic Collaborate, make sure you have met the requirements described in "Studio Configuration Requirements" on page 1-23.

Starting on Windows

To start WebLogic Process Integrator Studio on Windows:

1. Choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→Process Integrator→Start Studio

   Note: The Studio installation, and the `studio.cmd` file that is invoked by the shortcut, must be modified as described in "Studio Configuration Requirements" on page 1-23.

   The `studio.cmd` executes in a command window, then, within a few moments, the Studio application window is displayed.

2. Log in as described in "Logging In and Connecting to the WebLogic Process Integrator Server" on page 1-22.

Starting on UNIX

To start WebLogic Process Integrator Studio on UNIX:

1. Go to the `processintegrator/bin` directory, then run the `studio.sh` command. For example, if you installed Studio in the default location you would enter the following:

   ```
   cd <BEA_Home>/wlintegration2.0/processintegrator/bin
   . /studio.sh
   ```

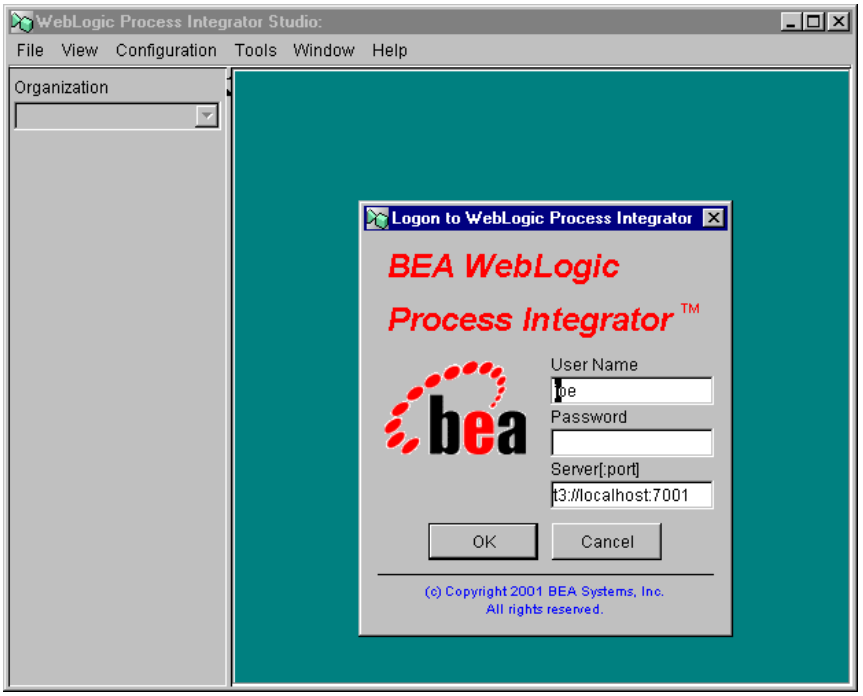
   The `studio.ch` command executes in a command window, then, within a few moments, the Studio application window is displayed.

2. Log in as described in "Logging In and Connecting to the WebLogic Process Integrator Server" on page 1-22.

Logging In and Connecting to the WebLogic Process Integrator Server

After you have executed the Studio command as described in the previous section, the logon dialog box is displayed in front of the WebLogic Process Integrator Studio application window as shown in the following figure.

Figure 1-8 Logon to WebLogic Process Integrator Dialog Box



To log on and connect to the WebLogic Process Integrator Server instance deployed in the active WebLogic Collaborate domain:

1. Enter the user name and password in the appropriate fields. If you have not yet been assigned a user name and password for WebLogic Process Integrator, enter a default user name and password. See “Understanding WebLogic Collaborate Users and Passwords” on page 1-4.

2. In the `Server [:port]` field, specify the system that is running the WebLogic Process Integrator Server application as follows:

   `t3://`*`host`*`:7001`

   Here, *`host`* is the computer name or IP address of the system that is running the WebLogic Process Integrator Server. Specify `localhost` if the server is running on the same computer as the Studio application.

Studio Configuration Requirements

If you have installed the Studio client on another system, or will be deploying WebLogic Collaborate in another domain, you need to be aware of the following requirements.

- The `fileRealm.properties` file for a domain that deploys WebLogic Collaborate, WebLogic Process Integrator, and the plug-in must have the access control entries required for the WebLogic Process Integrator users.

  Note: The `collaborate/config/mydomain/fileRealm.properties` and `collaborate/config/samples/fileRealm.properties` file include the required entries for the default users.

- The `wlc.jar` file must be added to the WebLogic Process Integrator Studio installation, and the `studio.cmd` (Windows) or `studio.sh` (UNIX) file must include the `wlc.jar` file in the CLASSPATH.

  Note: The CLASSPATH in the `collaborate/config/samples/studio.cmd` (or `.sh`) and `collaborate/config/mydomain/studio.cmd` (or `.sh`) file include the `wlc.jar` file.

Adding the Required Users to the fileRealm.properties File

The `fileRealm.properties` file is located in the root of the domain. The required entries for each user are:

```
acl.access.weblogic.admin.mbean.MBeanHome=user
acl.lookup.weblogic.admin.mbean.MBeanHome=user
```

To edit the `fileRealm.properties` file:

1. Do one of the following:
   - On Windows NT, Choose Start→Programs→Windows Explorer, then navigate to the root of the domain.
   - On UNIX, go to the root of the domain. For example, if you have set up a development domain, `devdomain`, in parallel with the custom and samples domain you would enter the following:

   ```
   cd <BEA_HOME>/bea/wlintegration2.0/collaborate/config/devdomain
   ```
2. Open the `fileRealm.properties` file in your preferred text editor.
3. Add the required entries.
4. Save your changes and close the file.

Adding wlc.jar to the CLASSPATH in the Start Studio Command

To add the `wlc.jar` file to the CLASSPATH in the `studio.cmd` (Windows) or `studio.sh` (UNIX) file:

1. Copy the `wlc.jar` file from the WebLogic Collaborate installation to the WebLogic Process Integrator Studio installation.

   The `wlc.jar` file is located in the `<BEA_Home>/wlintegration2.0/collaborate/lib` directory and should be copied to the `<BEA_Home>/wlintegration2.0/processintegrator/lib` directory for the Studio installation.
2. Do one of the following:
   - On Windows NT, Choose Start→Programs→Windows Explorer, then navigate to the `bin` subdirectory of the WebLogic Process Integrator Studio installation.
   - On UNIX, go to the `bin` subdirectory of the WebLogic Process Integrator Studio installation. For example, if you installed Studio in the default location, you would enter the following:

   ```
   cd <BEA_HOME>/bea/wlintegration2.0/processintegrator/bin
   ```

3. Open the `file.properties` file in your preferred text editor.
4. Add the complete path to the `wlc.jar` file to the CLASSPATH. For example, if you installed Studio on a Windows system in the default location, you would add the following to the CLASSPATH, using a semi-colon to separate this entry from others in the CLASSPATH:

   ```
   c:/bea/wlintegration2.0/processintegrator/lib/wlc.jar
   ```

5. Save your changes and close the file.

Customizing WebLogic Collaborate

The following sections describe how to modify the default installation.

- About Updating the JDBC Configuration
- Updating the JDBC Connection Pool
- Updating the WebLogic Collaborate Environment
- Updating the Default Passwords

About Updating the JDBC Configuration

As discussed in "Understanding the Preconfigured Domains" on page 1-2, when you installed WebLogic Collaborate, the JDBC connection pool for the samples domain is configured to use the default Cloudscape database. Depending on the information provided during installation, the JDBC connection pool for the custom domain was configured to use either:

- The same Cloudscape database as the samples domain
- An Oracle or Microsoft SQL Server database of your choice

In a standard installation, each WebLogic Integration component follows this model, with a samples domain configured to use the default Cloudscape domain, and a custom domain configured to use either the Cloudscape or the alternate database. Therefore, the components share the database resources.

Although this arrangement typically provides an environment that meets your evaluation, development, and testing needs, there are circumstances that may require you to customize the JDBC configuration. For example, you may wish to:

- Run the samples in the alternate database you have already configured for the custom domain.
- Create and connect to a new database.

The first case is fairly straightforward. As described in *Using BEA WebLogic Collaborate Samples*, commands have been provided to assist you in reconfiguring the the samples domain to use the alternate database you selected during installation.

The second case, where it becomes necessary to change the database used (for example, from Microsoft SQL to Oracle) you can do one of the following:

- Install WebLogic Server and the WebLogic Integration components in a new location, and specify the required database during installation. (Recommended.)
- Update the JDBC connection pool for one or more domains using the WebLogic Server Administration Console as described "Updating the JDBC Connection Pool" on page 1-27.

Installing the WebLogic Integration components in a new location is the recommended method. When this methods is used, the JDBC connection pool is automatically configured as part of the installation process. Because you are not overwriting the existing installation, the information remains available and can be exported as needed for import to the new installation.

If you choose the second method, keep the following in mind:

- The WebLogic Integration components share the database resources. You must be sure that the tables required for each installed component are recreated in the new database. Each component distribution includes the script(s) required to create the tables it needs. Be sure to execute each script as instructed in the appropriate installation guide.
- The WebLogic Process Integrator RDBMS security realm uses the same database. In addition to updating the JDBC connection pool for WebLogic Collaborate, you must update the RDBMS realm properties for WebLogic Process Integrator. See "Updating the RDBMS Realm Properties" on page 1-29.

- Export any information you may need to transfer to the new installation *before* making the change. To understand what is stored in the WebLogic Collaborate repository, see Chapter 7, "Working with the Repository." For instructions on exporting the WebLogic Process Integrator workflows and WebLogic Collaborate elements required for business collaborations, see Chapter 5, "Importing and Exporting Business Collaborations." Store the exported files in a secure location for later import to the new database.

Updating the JDBC Connection Pool

To update the WebLogic Integration JDBC connection pool:

1. Start the WebLogic Server Administration Console as described in "Starting the WebLogic Server Administration Console" on page 1-17.
2. Using the navigation tree, choose Services→JDBC→Connection Pools→wlipool.
3. Select the Configuration tab. Then, if it is not already displayed, select the General tab.
4. Edit the URL, Driver Classname, and Properties as required to customize the settings for your JDBC connection pool. For additional information see "JDBC Connection Pool Parameters" in the following section.
5. Click Apply to save your changes.
6. Close the WebLogic Server Administration Console.
7. Shut down and restart WebLogic Collaborate to initiate the new settings.

JDBC Connection Pool Parameters

The following table summarizes the information required to configure the JDBC connection pool.

Table 1-2 JDBC Connection Pool Parameters

Parameter	Description
JDBC Connection Pool Driver	JDBC driver to be used to connect to the database. ■ If you are using the Oracle thin driver, enter: `oracle.jdbc.driver.OracleDriver` ■ If you are using the Sybase jConnect driver, enter: `com.sybase.jdbc.SybDriver` ■ If you are using the Cloudscape driver, enter: `COM.cloudscape.core.JDBCDriver` ■ If you are using the WebLogic jDriver for SQL Server, enter: `weblogic.jdbc.mssqlserver4.Driver`
JDBC Connection Pool User	Account login name required for connecting to the database server.
JDBC Connection Pool Password	Password required for connecting to the database server.
JDBC Connection Pool URL	URL for the database, as specified in the JDBC driver documentation. The format for the JDBC connection pool URL is discussed in the following section.

JDBC Connection Pool URL Format

The JDBC connection pool URL includes the following:

- *`host`* specifies the name of the database server host.
- *`database`* specifies the database containing the WebLogic Process Integrator tables on the database server.
- *`port`* specifies the port to be used to connect to the database server.

The following list provides a sample URL for each supported database.

- For the Oracle thin driver, the URL is:

  `jdbc:oracle:thin:@`*`host`*`:`*`port`*`:`*`database`*

  For example: `jdbc:oracle:thin:@rdbmshost:1521:wlidb`

- For the Cloudscape driver, the URL is:

  `jdbc:cloudscape:`*`database`*

  For example: `jdbc:cloudscape:wlidb`

- For the WebLogic jDriver for Microft SQL Server, the URL is:

  `jdbc:weblogic:mssqlserver4:`*`database`*`@`*`host`*`:`*`port`*`?sql7=true`

  For example:
  `jdbc:weblogic:mssqlserver4:wlidb@rdbmshost:1433?sql7=true`

Updating the RDBMS Realm Properties

To update the WebLogic Process Integrator RDBMS realm properties:

1. Start the WebLogic Server Administration Console as described in "Starting the WebLogic Server Administration Console" on page 1-17.
2. Using the navigation tree, choose Security→Realms→wlpiRDBMSRealm.
3. Select the Configuration tab.
4. Select the Database tab.
5. Edit the Driver, URL, Username, and Password as required to customize the settings for your database. For additional information see "JDBC Connection Pool Parameters" in the preceding section.
6. Click Apply to save your changes.
7. Close the WebLogic Server Administration Console.
8. Shut down and restart WebLogic Collaborate to initiate the new settings.

Updating the WebLogic Collaborate Environment

The WebLogic Collaborate environment is set by the `setEnv.cmd` (Windows) or `setEnv.sh` (UNIX) file. This file is located in the `config\domain_name` subdirectory, as described in "Understanding the Preconfigured Domains" on page 1-2. An example of this file is provided in Appendix A, "WebLogic Collaborate Sample Configuration Files."

The variables in this file are set when you install WebLogic Collaborate and normally do not require update. If you must update the environment, however, you can do so by completing the following procedure.

To update the WebLogic Collaborate environment:

1. Do one of the following:
   - On a Windows system:

     Navigate to the appropriate domain directory and right click `setEnv.cmd`. Then select Edit from the shortcut menu to open the file in Notepad.
   - On a UNIX system.

     Go to the appropriate domain directory, and open `setEnv.sh` in your preferred text editor.
2. Set the following variables to values appropriate for your environment:
   - `JAVA_HOME`
   - `BEA_HOME`
   - `WL_HOME`
   - `WLC_HOME`
   - `WLPI_HOME`

   For information about the required values, see Appendix A, "WebLogic Collaborate Sample Configuration Files."
3. Save your changes and close the file.

The `setEnv.cmd` file will run to set these environment variables for WebLogic Collaborate when you execute the start command as described in "Starting WebLogic Collaborate" on page 1-5.

Updating the Default Passwords

After you have installed WebLogic Collaborate, you can update the default passwords to ensure system security.

Warning: The `wlcsystem` user name and password are used by the WebLogic Collaborate run-time environment and are stored in the WebLogic Collaborate repository. This password *must* be updated using the WebLogic Collaborate Administration Console. *Do not* use the WebLogic Server Administration Console to update this password.

Updating the system Password

The password for the `system` login can be changed through the WebLogic Server Administration Console as described in the following procedure.

To update the password for the `system` user for the active domain:

1. Start the WebLogic Server Administration Console as described in "Starting the WebLogic Server Administration Console" on page 1-17.
2. Select Users from the navigation tree to open the Users page.
3. In the Change a User's Password section, enter the user name in the Name field.

   For example, to change the system password, enter `system`.
4. Enter the existing password in the a Old Password field.
5. Enter the new password in the New Password field.
6. Retype the new password in the Confirm Password field.
7. Click Change to update the password.
8. Update the `startWebLogic.cmd` (Windows) or `startWebLogic.sh` (UNIX) for the domain.

   The `startWeblogic.cmd` (Windows) or `startWeblogic.sh` (UNIX) commands used to start WebLogic Collaborate are configured for automatic login. If you change the password for `system`, you will need to update these files. See Appendix A, "WebLogic Collaborate Sample Configuration Files."

Updating the default WebLogic Process Integrator Passwords

The default passwords for `admin`, `joe`, and `mary`, which are used to log on to WebLogic Process Integrator Studio and Worklist, can also be changed through the WebLogic Server Administration Console as described in the previous section.

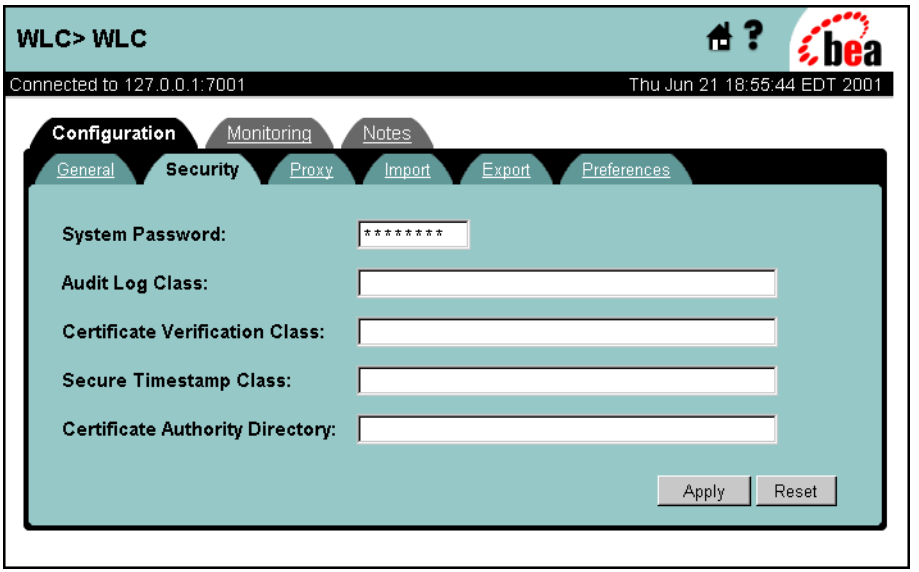
Updating the wlcsystem Password

The password for the `wlcsystem` user can only be changed through the WebLogic Collaborate Administration Console as described in the following procedure.

1. Start the WebLogic Collaborate Administration Console as described in "Starting the WebLogic Collaborate Administration Console" on page 1-7.
2. Select WebLogic Collaborate (WLC) from the navigation tree.
3. If it is not already selected, select the Configuration tab, then select the Security tab.

   The Security tab is displayed as shown in the following figure. The content of the System Password field is the password for the `wlcsystem` user.

Figure 1-9 WebLogic Collaborate Configuration Security Tab



4. To update the `wlcsystem` password, highlight the current content of the System Password field, then carefully overtype with the new password.
5. Click Apply to update the `wlcsystem` password.

Verifying Configuration

If you make changes to the WebLogic Collaborate installation, or configure your own domain, you can verify the WebLogic Collaborate configuration by starting WebLogic Collaborate for the domain, then starting the WebLogic Collaborate Administration Console for the domain as described in this section.

We also recommend that you verify the configuration of the samples domain by running the Hello Partner sample as described in *Using BEA WebLogic Collaborate Samples*. In order to run the Hello Partner sample, you will need to load the sample data and workflows into the WebLogic Collaborate repository as described in "Hello Partner Sample" in *Using BEA WebLogic Collaborate Samples*.

2 Configuration Requirements

This section presents selected WebLogic Collaborate 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 material in this section applies to configurations that use WebLogic Process Integrator workflows to manage conversations between trading partners. If you are using the WebLogic Collaborate API to develop applications to exchange business messages, see *Programming BEA WebLogic Collaborate Messaging Applications*.

If you are using logic plug-ins to add functionality to a collaboration, see *Programming BEA WebLogic Collaborate Logic Plug-Ins*.

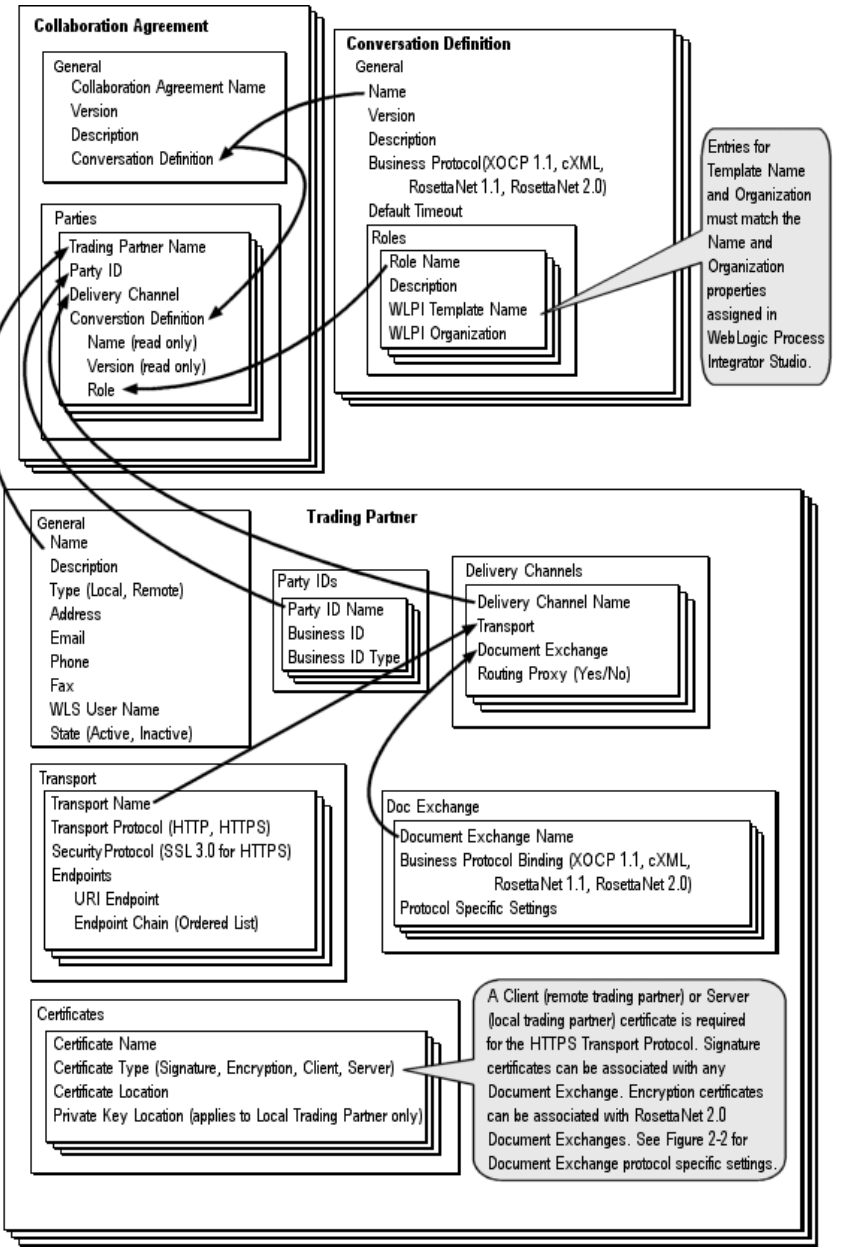
Configuration Overview

The sections that follow describe what you need to know to set up some of the basic configurations supported by WebLogic Collaborate. These basic configurations illustrate the concepts required for more complex scenarios. In addition to basic WebLogic Collaborate parameter settings (for example, server name, description, and preferences), the following must be configured:

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

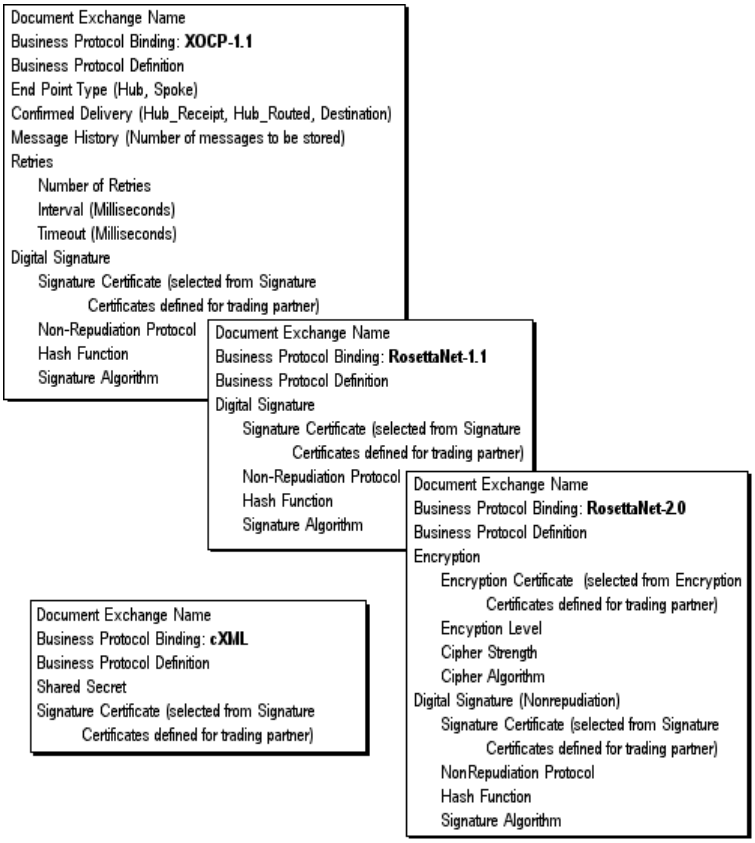
The following figure provides an overview of the configuration and the relationships between the entities. The view presented illustrates how the information is organized in the WebLogic Collaborate Administration Console. For a summary of how the information is stored in the repository, see Chapter 7, “Working with the Repository.”

Figure 2-1 WebLogic Collaborate Configuration Overview



As shown in the preceding figure, each delivery channel is associated with a document exchange, which defines the business protocol binding for the delivery channel. The settings available are dependent on the selected protocol. The following figure summarizes each type.

Figure 2-2 Document Exchange Business Protocol Bindings



In the sections that follow, the configuration requirements for participants in selected example applications are presented. These examples do not take into account how the required elements were added to the WebLogic Collaborate configuration. Keep in mind that the required 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 5, “Importing and Exporting Business Collaborations.”

XOCP Applications

The following sections provide background information about the configuration of XOCP delivery channels and present examples that illustrate how to configure WebLogic Collaborate 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 either a:

- A routing proxy, or hub, delivery channel.
- A standard, or 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 is assigned to the role of buyer in another collaboration agreement for the same conversation definition, messages will be handled as follows:

- The hub delivery channel for a trading partner receives a message to the role supplier. Any collaboration agreements for the same conversation definition where the hub delivery channel for that trading partner is assigned the role buyer are identified. The message is forwarded to the trading partner delivery channel assigned to the role supplier in each collaboration agreement identified.

- The hub delivery channel for a trading partner receives a message to the role buyer. Any collaboration agreements for the same conversation definition where the hub delivery channel for that trading partner is assigned the role supplier are identified. The message is forwarded to the trading partner delivery channel assigned to the role buyer in each collaboration agreement identified.

As can be seen, 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 4-3.

Although XOCP applications always adhere to a hub-and-spoke configuration with regard to the configuration of 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***
  In this case, each participating trading partner has a role in the collaboration. The messages are exchanged between the trading partner peers. To accomplish this, at least one of the trading partner peers must configure two delivery channels. One channel is configured as a hub delivery channel; the other, as a spoke delivery channel.

- ***Mediated***
  In this case, a trading partner mediator does not directly participate in any role, but rather serves to route messages to the trading partner spokes. The trading partner acting as 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 agreements assign the hub and spoke delivery channels to roles as required to properly route the messages.

Detailed examples illustrating these two basic XOCP arrangements are described in the following sections. The principles illustrated in the examples can be employed 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 Collaborate to participate in Query Price and Availability (QPA) transactions. ABC International will be the buyer and the initiator of each transaction, and the XOCP protocol will be used to exchange messages. Both trading partners have WebLogic Collaborate installed.

The sections that follow provide an example 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 in collaboration agreements

It is assumed that the required private and collaborative (public) workflows have been created. For the purposes of this example, the collaborative workflows are named `QPA_Public_Supplier` and `QPA_Public_Buyer`. For information about using the WebLogic Collaborate plug-in with WebLogic Process Integrator to create collaborative workflows, see *Creating Workflows for BEA WebLogic Collaborate*.

Trading Partners

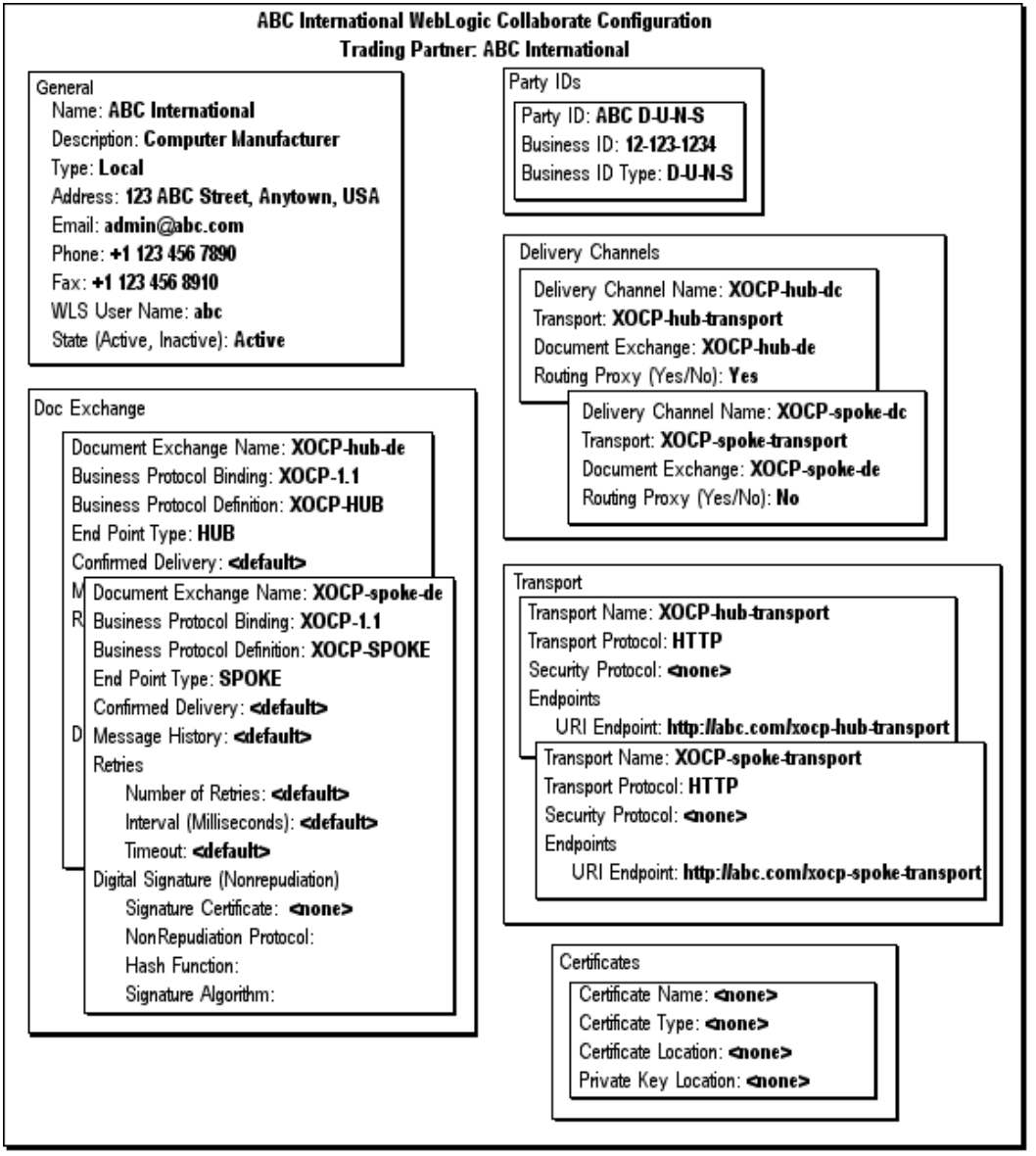
Trading partner definitions for both ABC International and XYZ Systems must be configured.

Note: For simplicity, SSL or signature certificates are not used in the examples presented in this section. For information about the security configuration requirements, see *Using BEA WebLogic Collaborate Security*.

The requirements for the ABC International trading partner in the ABC WebLogic Collaborate configuration are summarized in the following figure.

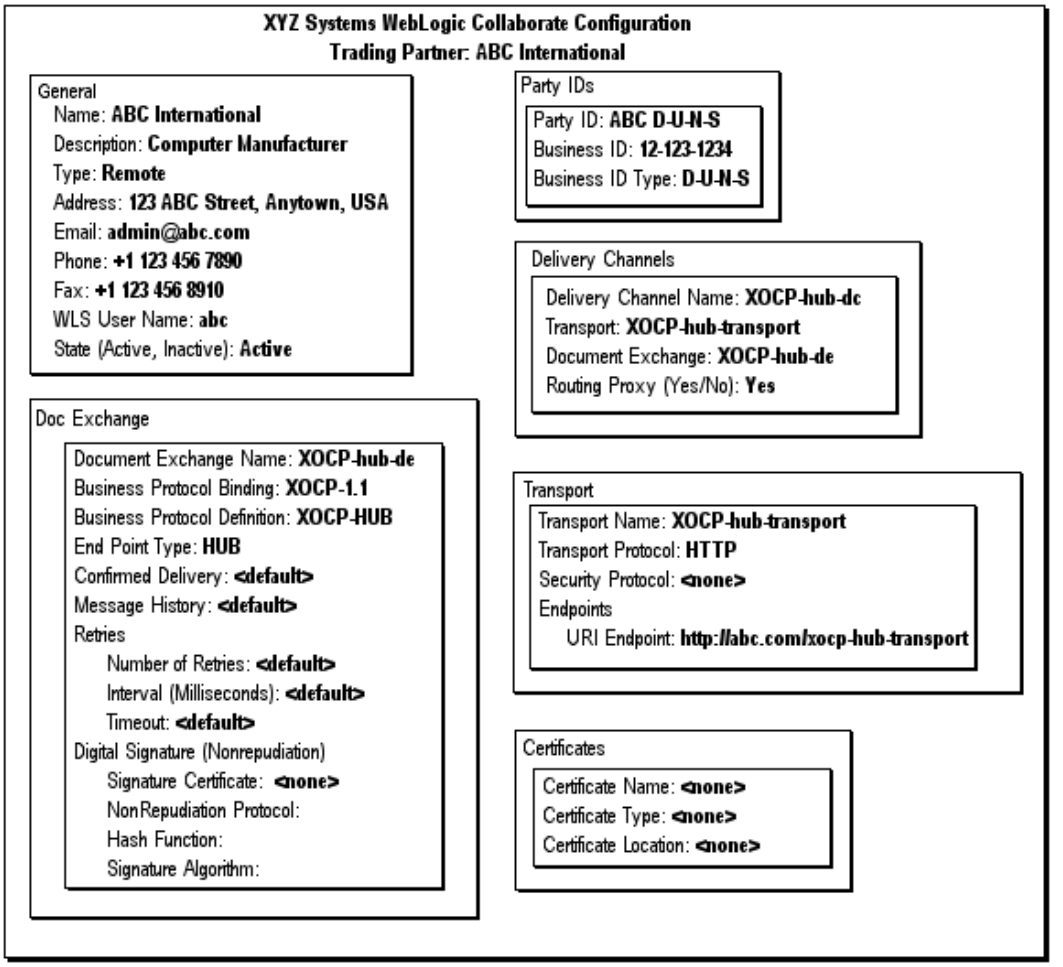
Note that the trading partner type is `Local` and the configuration includes hub and spoke delivery channels (`XOCP-hub-dc` and `XOCP-spoke-dc`), document exchanges (`XOCP-hub-de` and `XOCP-spoke-de`), and transports (`XOCP-hub-transport` and `XOCP-spoke-transport`). Each transport has an associated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

Figure 2-3 ABC International Trading Partner Definition in the ABC Configuration



The requirements for the ABC International trading partner in the XYZ WebLogic Collaborate configuration are summarized in the following figure.

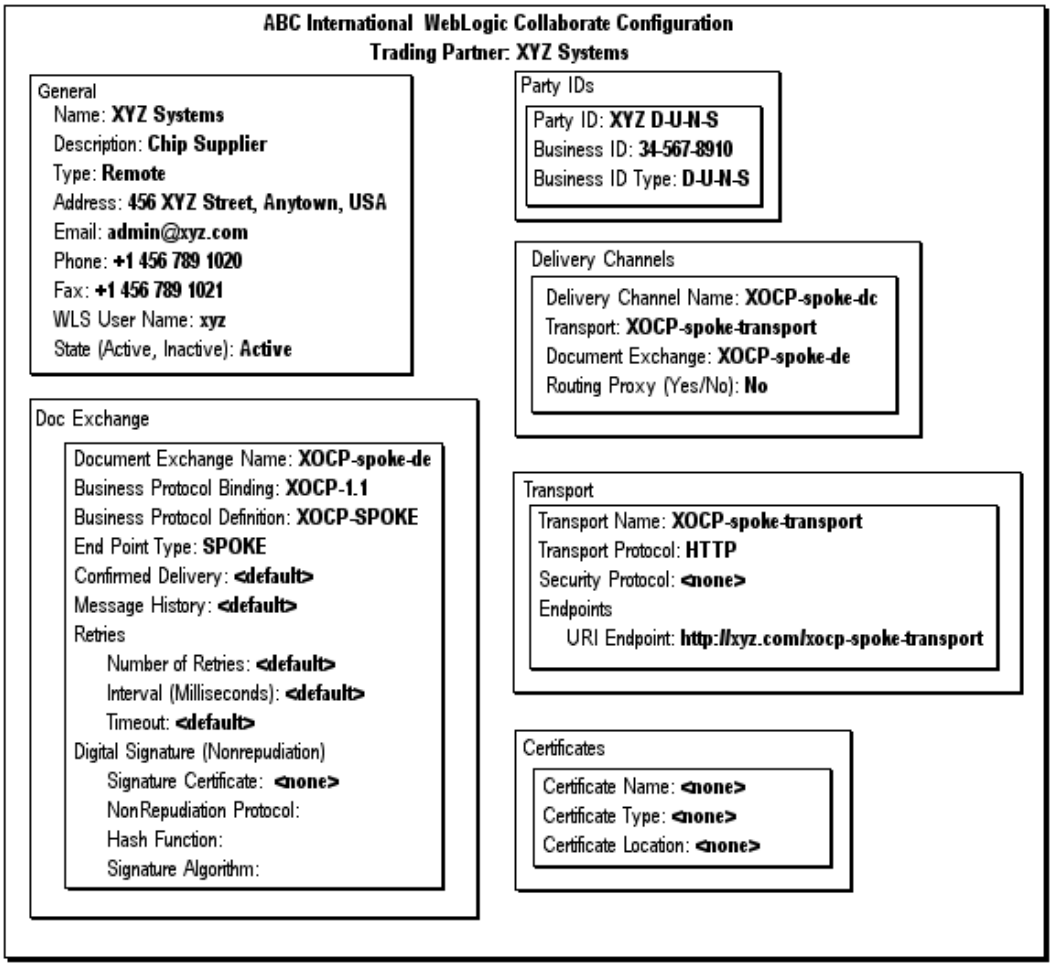
Figure 2-4 ABC International Trading Partner Definition in the XYZ Configuration



In this case the trading partner type 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 has an associated URI which serves as the delivery channel endpoint (`http://abc.com/xocp-hub-transport`).

The requirements for the XYZ Systems trading partner in the ABC WebLogic Collaborate configuration are summarized in the following figure.

Figure 2-5 XYZ Systems Trading Partner Definition in the ABC International Configuration



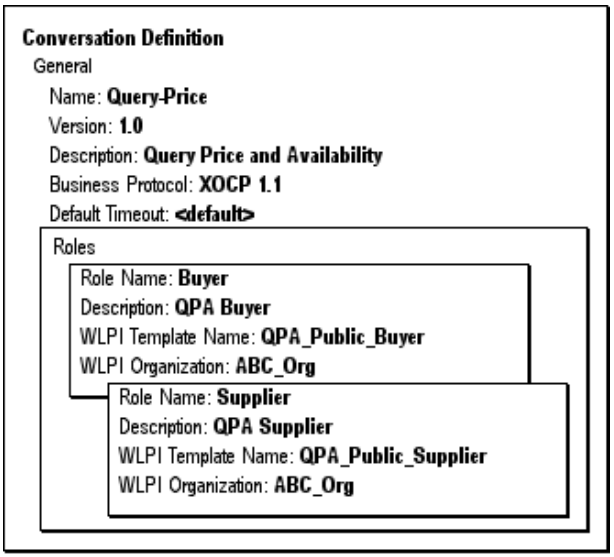
In this case the trading partner type is `Remote` and the configuration includes only a spoke delivery channel (`XOCP-spoke-dc`), document exchange (`XOCP-spoke-de`), and transport (`XOCP-spoke-transport`). The transport has an associated URI which serves as the delivery channel endpoint (`http://xyz.com/xocp-spoke-transport`).

The requirements for the XYZ Systems trading partner in the XYZ WebLogic Collaborate configuration are exactly the same as those just listed, with the exception of trading partner type. Trading partner type is set to `Local` in the XYZ WebLogic Collaborate configuration.

Conversation Definitions

The requirements for the Query Price and Availability (QPA) conversation definition in the ABC WebLogic Collaborate configuration are summarized in the following figure.

Figure 2-6 Conversation Definition for QPA

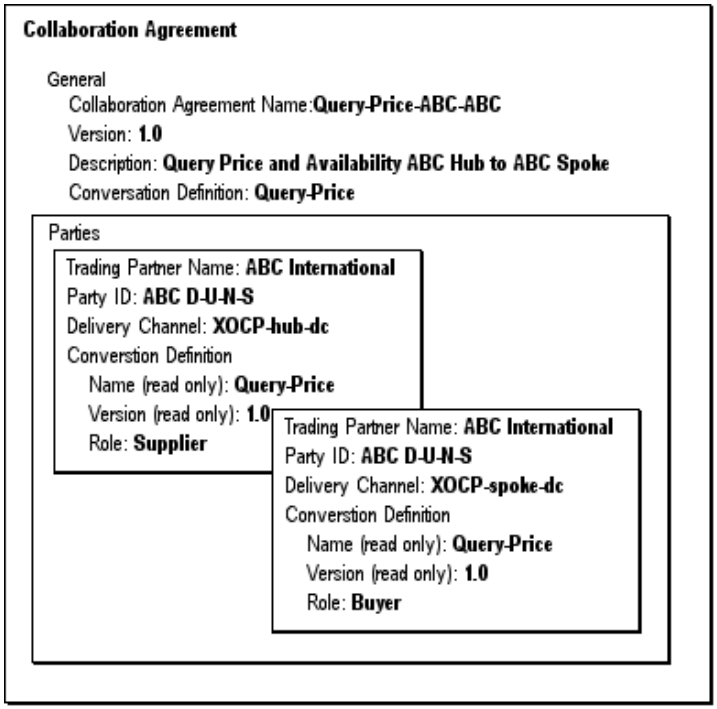


The conversation definition in the XYZ WebLogic Collaborate configuration is the same, with the exception of the WebLogic Process Integrator (WLPI) organization, which reflects the structure used at XYZ Systems.

Collaboration Agreements

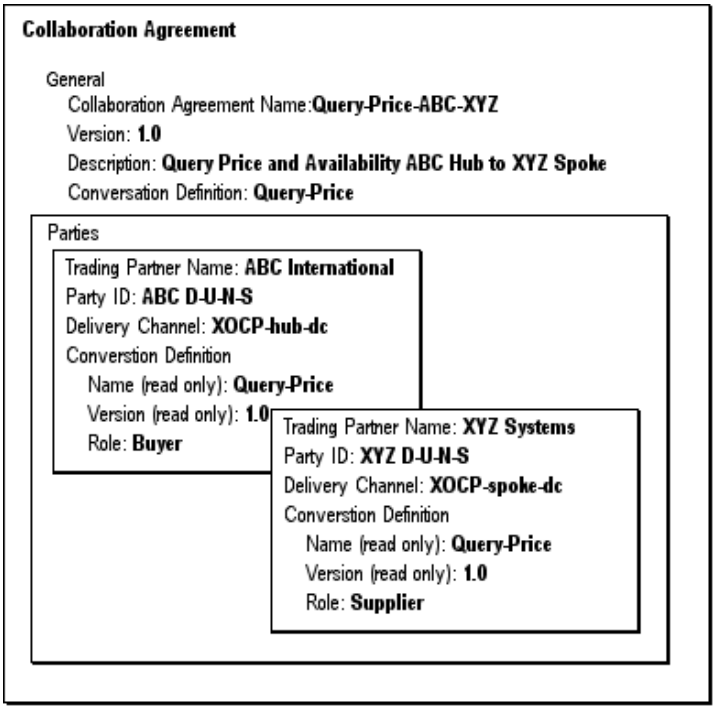
The collaboration agreement shown in the following figure is required only in the ABC WebLogic Collaborate configuration. The agreement makes it possible for the hub delivery channel to act as a proxy for the buyer role, as described in "XOCP Hub and Spoke Delivery Channels" on page 2-5.

Figure 2-7 Collaboration Agreement ABC-ABC



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

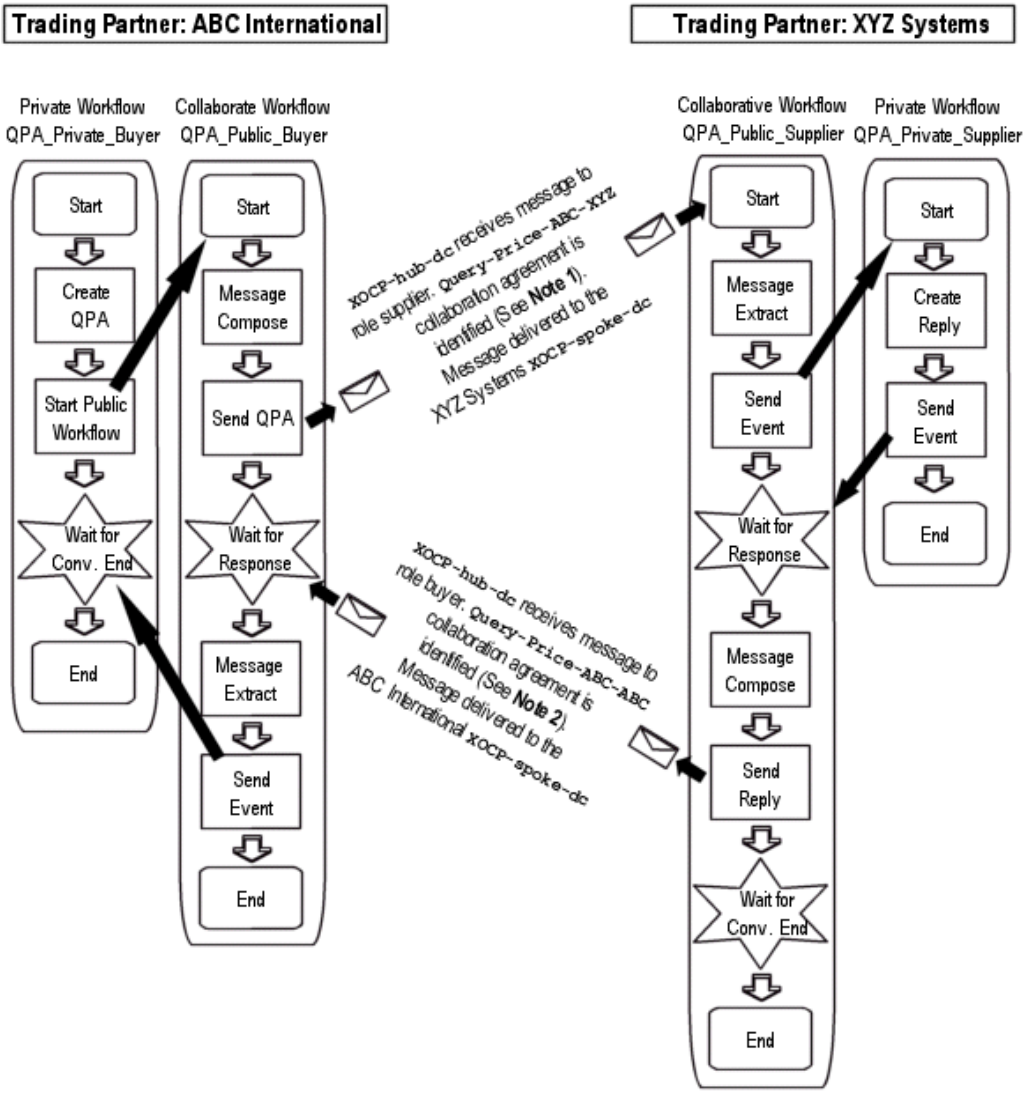
Figure 2-8 Collaboration Agreement ABC-XYZ



How It Works

The following figure summarizes how WebLogic Collaborate supports the exchange of Query Price and Availability (QPA) messages between ABC International and XYZ Systems.

Figure 2-9 QPA Collaboration Overview: XOCP Peer-to-Peer



Note 1
The ABC International hub delivery channel (`XOCP-hub-dc`) receives the message to the supplier role. As described in "XOCP Hub and Spoke Delivery Channels" on page 2-5, any collaboration agreements in which `XOCP-hub-dc` is assigned the role buyer are identified. 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 to the buyer role. Any collaboration agreements in which `XOCP-hub-dc` is assigned the role of supplier are identified. 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 will act 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 rather acts as a mediator in the transactions between ABC International and the suppliers.

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

The sections that follow provide an example 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 in collaboration agreements.

It is assumed that the required private and collaborative (public) workflows have been created. For the purposes of this example, the collaborative workflows are named `QPA_Public_Supplier` and `QPA_Public_Buyer`. For information on using the WebLogic Collaborate plug-in with WebLogic Process Integrator to create collaborative workflows, see *Creating Workflows for BEA WebLogic Collaborate*.

Trading Partners

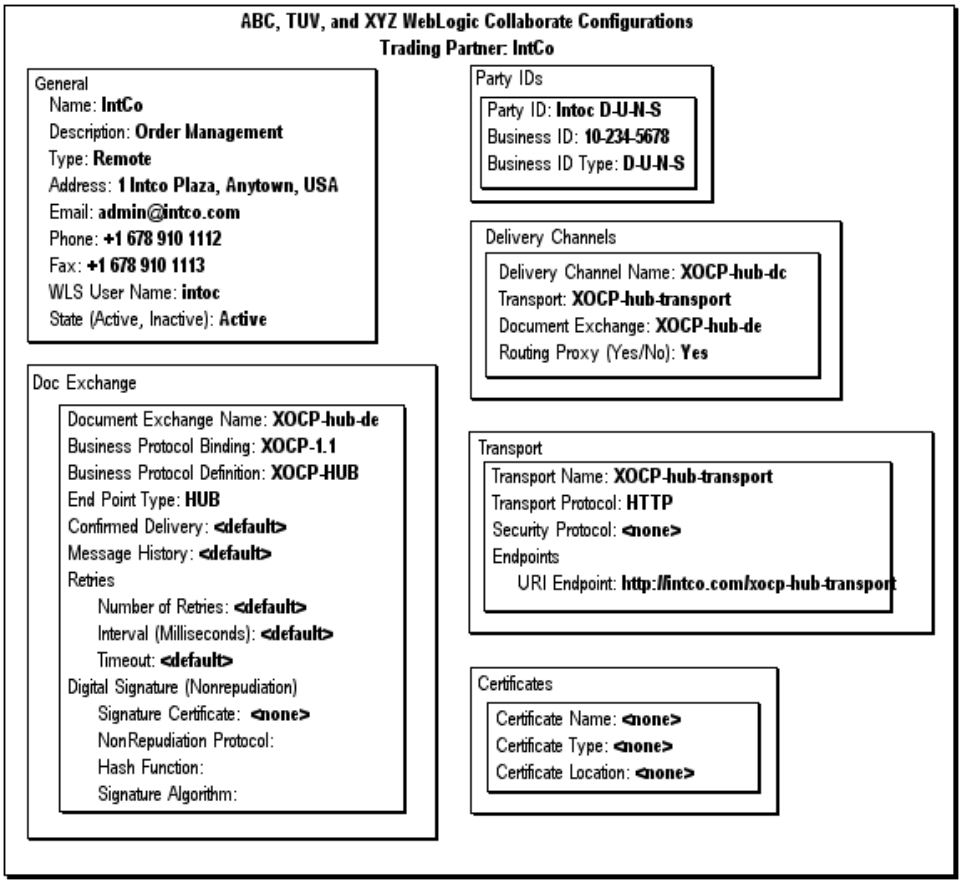
Trading partners must be configured as follows:

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

Note: For simplicity, SSL or signature certificates are not used in the examples presented in this section. For information about the security configuration requirements, see *Using BEA WebLogic Collaborate Security*.

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

Figure 2-10 IntCo Trading Partner Definition in the ABC, TUV, and XYZ Configurations



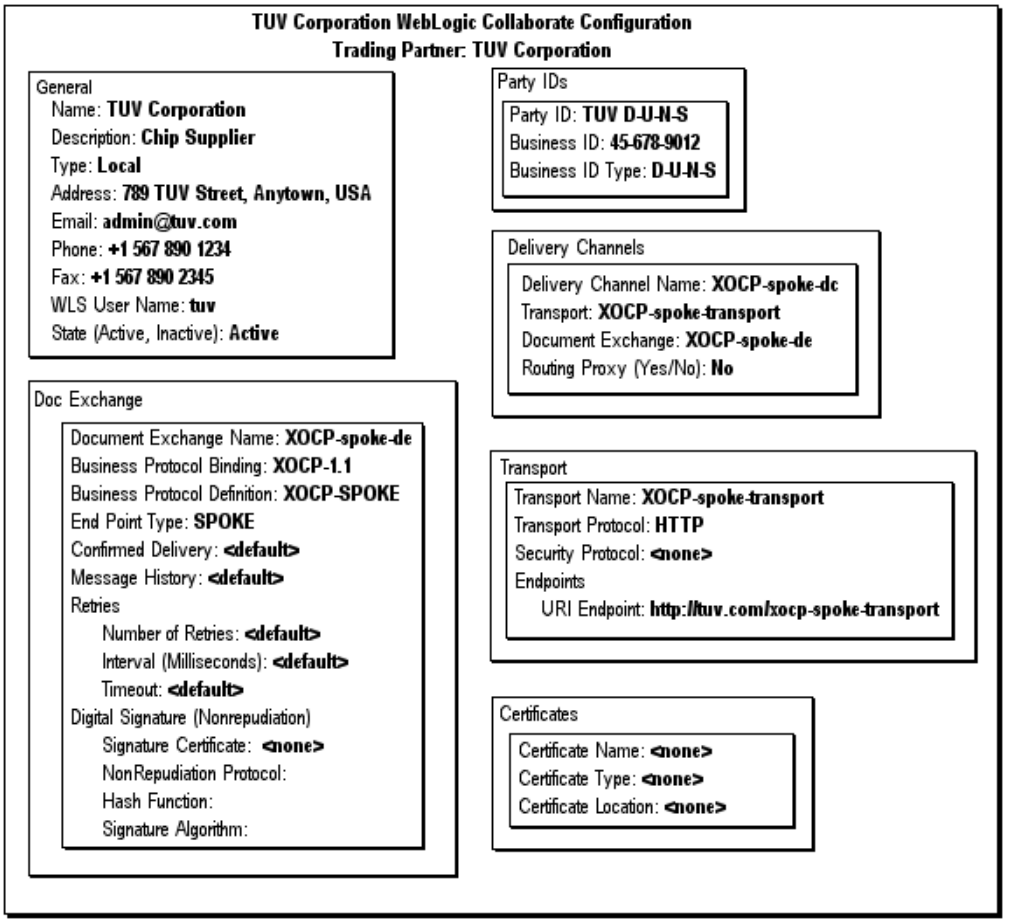
Note that the trading partner type is `Remote` and the configuration includes a hub delivery channel (`XOCP-hub-dc`), document exchange (`XOCP-hub-de`), and transport (`XOCP-hub-transport`). The transport has an associated URI which serves as the delivery channel endpoint (`http://intco.com/xocp-hub-transport`).

The requirements for the IntCo trading partner definition in the IntCo WebLogic Collaborate configuration are exactly the same as those just listed, with the exception of trading partner type. Trading partner type is set to `Local` in the IntCo WebLogic Collaborate configuration.

ABC International, TUV Corporation, and XYZ Systems must each configure a trading partner definition for themselves.

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

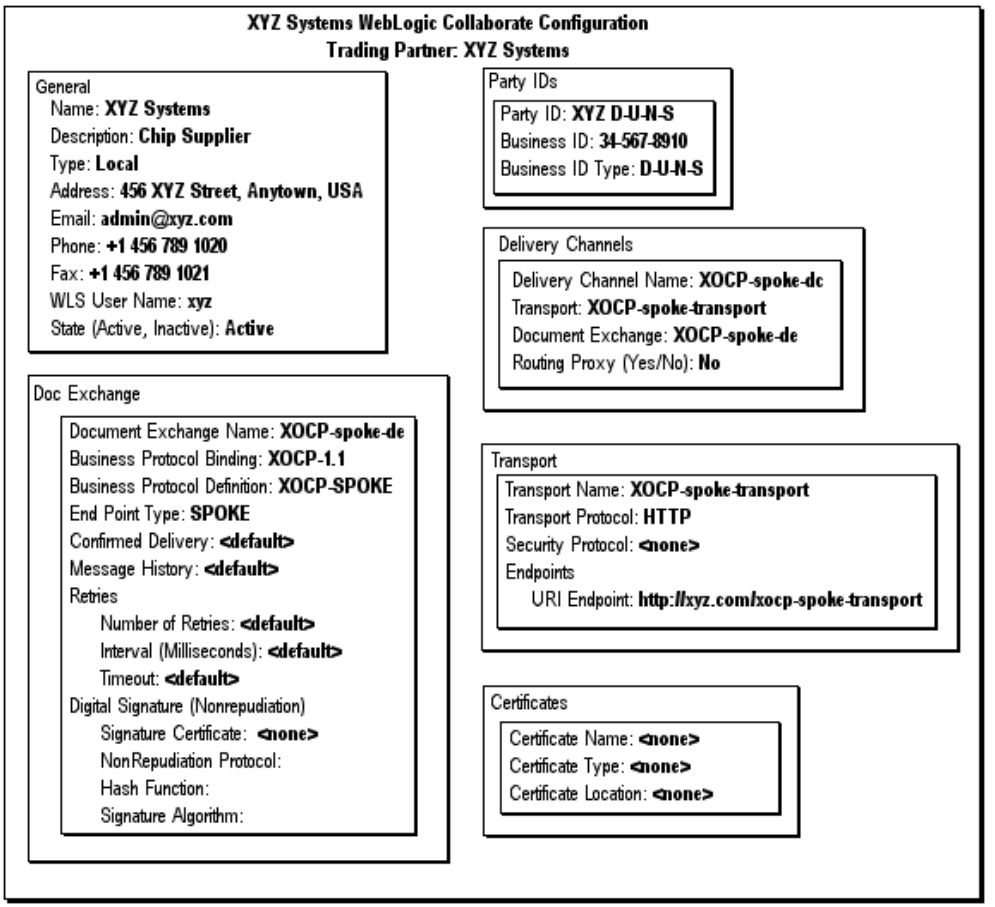
Figure 2-11 TUV Corporation Trading Partner Definition in the TUV Configuration



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 has an associated URI which 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.

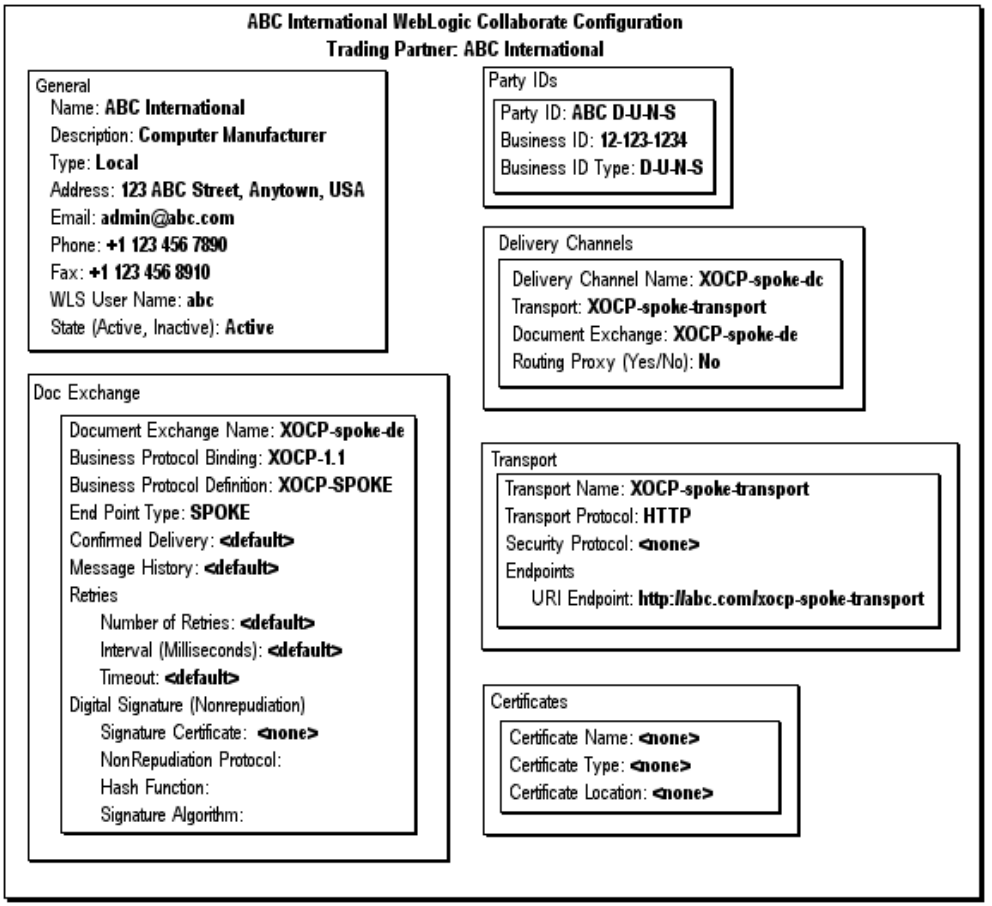
Figure 2-12 XYZ Systems Trading Partner Definition in the XYZ Configuration



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 has an associated URI which 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 2-13 ABC International Trading Partner Definition in the ABC Configuration



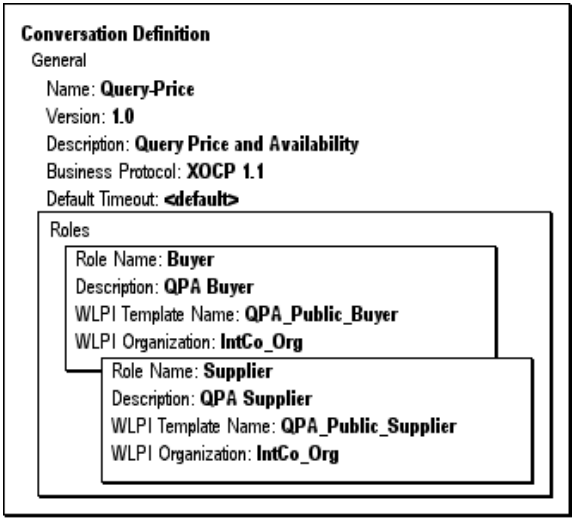
Note that the trading partner type is `Local` and the configuration includes a spoke delivery channel (`XOCP-spoke-dc`), document exchange (`XOCP-spoke-de`), and transport (`XOCP-spoke-transport`). The transport has an associated URI which serves as the delivery channel endpoint (`http://abc.com/xocp-spoke-transport`).

The requirements for the XYZ, TUV, and ABC trading partner definitions in the IntCo WebLogic Collaborate 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 Collaborate configuration.

Conversation Definitions

The requirements for the Query Price and Availability (QPA) conversation definition in the IntCo WebLogic Collaborate configuration are summarized in the following figure.

Figure 2-14 Conversation Definition for QPA

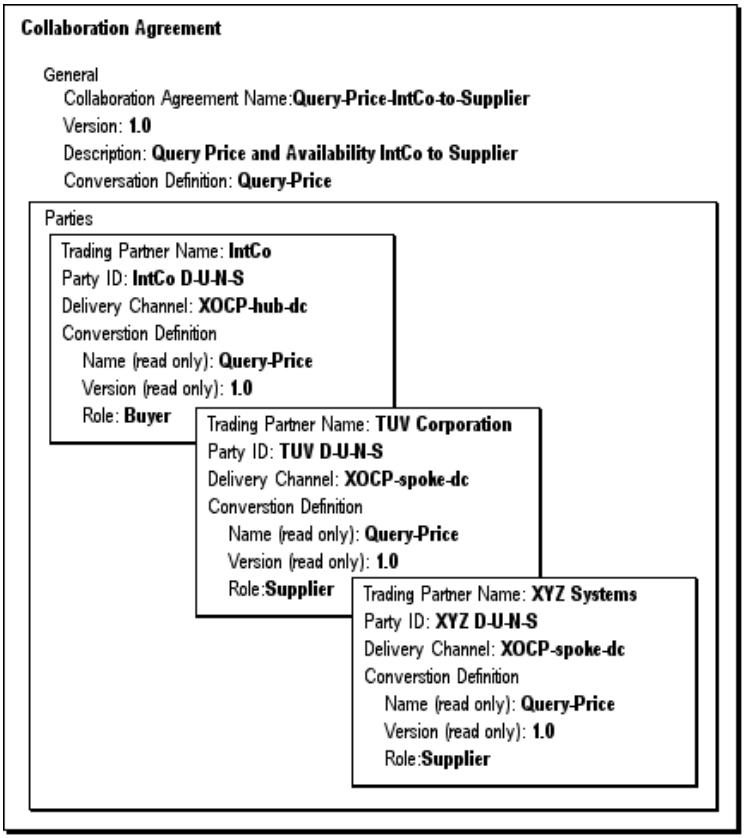


The conversation definition in the ABC, TUV, and XYZ WebLogic Collaborate configurations is the same, with the exception of the WebLogic Process Integrator (WLPI) organization, which reflects the structure used at each respective company.

Collaboration Agreements

The collaboration agreement shown in the following figure is required in the IntCo WebLogic Collaborate configuration to allow 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 2-5.

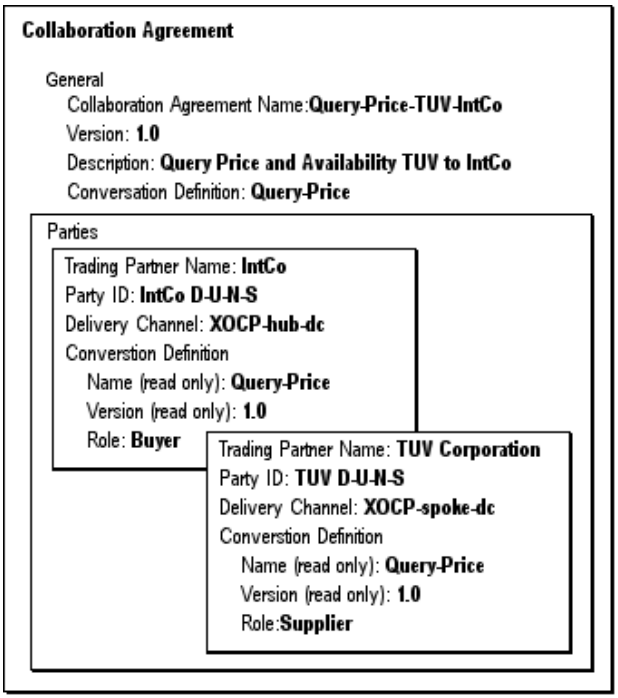
Figure 2-15 Collaboration Agreement IntCo-to-Supplier



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.

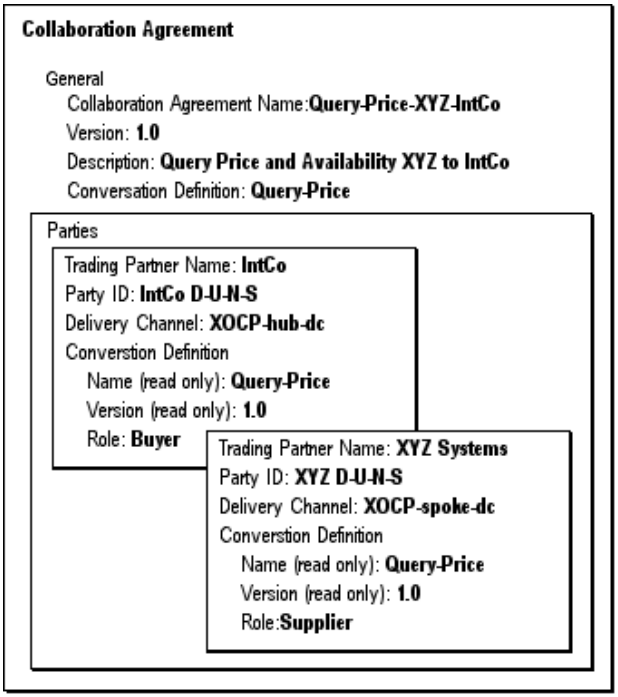
Corresponding agreements are required in the TUV and XYZ configurations. For example, the collaboration agreement required in the TUV configuration is shown in the following figure.

Figure 2-16 Collaboration Agreement TUV-IntCo



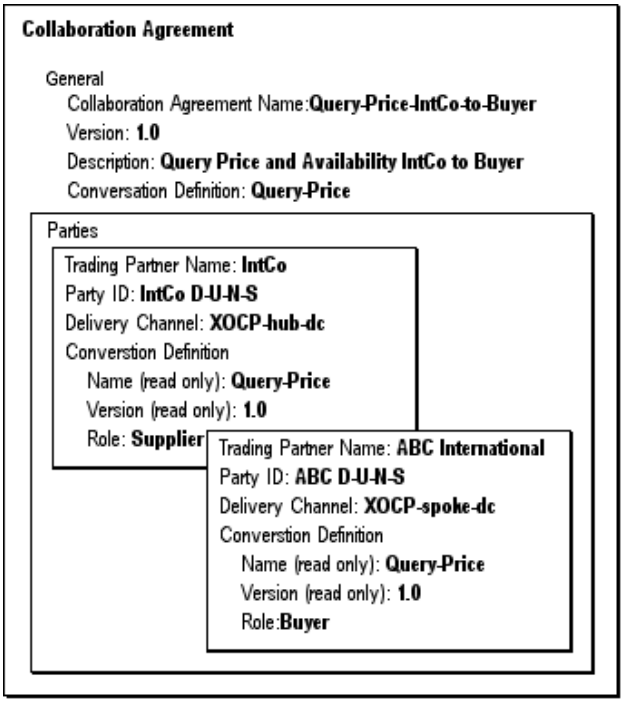
The agreement required in the XYZ configuration is shown in the following figure.

Figure 2-17 Collaboration Agreement XYZ-IntCo



The collaboration agreement shown in the following figure is required in the IntCo WebLogic Collaborate configuration to allow 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 2-5.

Figure 2-18 Collaboration Agreement IntCo-to-Buyer

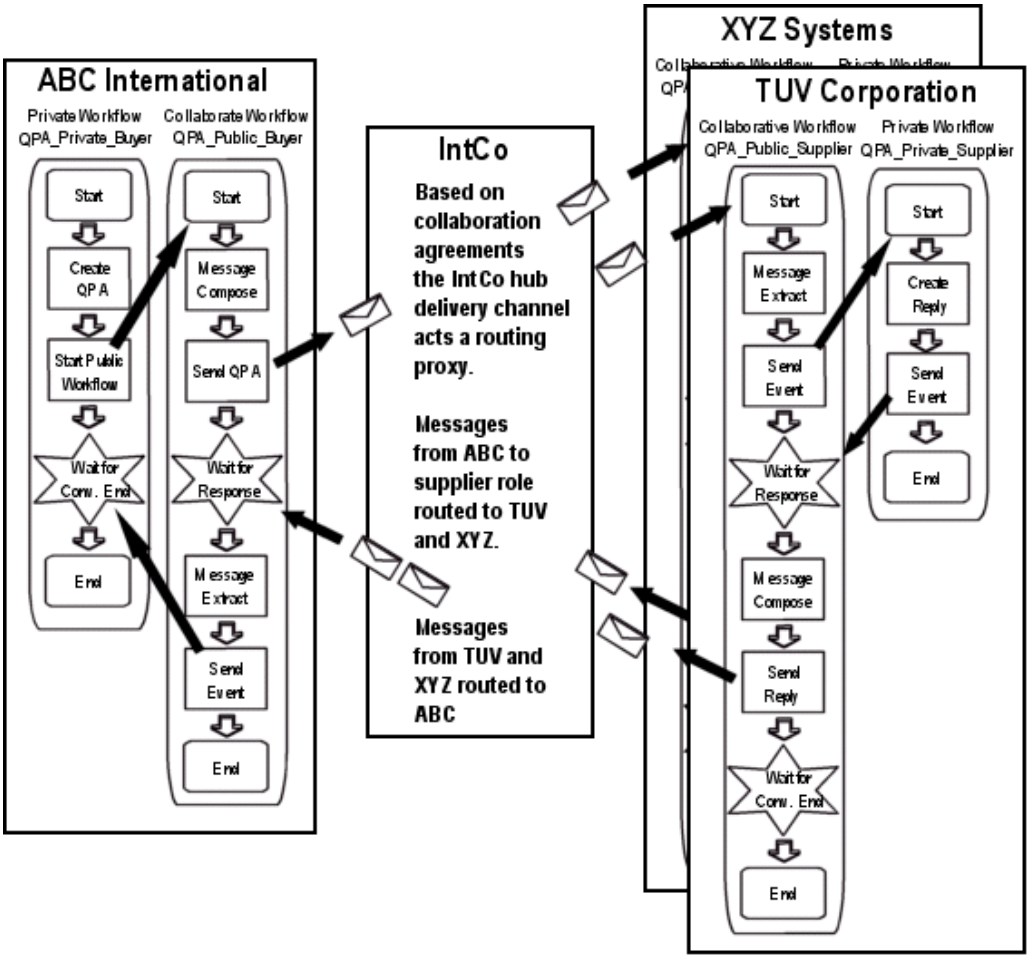


The agreement shown in the preceding figure is also required in the ABC configuration.

How It Works

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

Figure 2-19 QPA Collaboration Overview: XOCP Mediated



The message sent to the supplier role by the `Send QPA` action of the ABC `QPA_Public_Buyer` workflow is sent to the IntCo hub delivery channel based on the collaboration agreement defined in the ABC configuration.

Upon receipt of the message to the supplier role on the IntCo hub delivery channel (`XOCP-hub-dc`), any collaboration agreement for the `Query-Price` conversation definition in which `XOCP-hub-dc` is assigned the role buyer are identified. One collaboration agreement, `Query-Price IntCo-to-Supplier`, meets the criteria. Based on 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 channel based on the collaboration agreement defined in the respective configurations.

Upon receipt of a message to the buyer role on the IntCo hub delivery channel (`XOCP-hub-dc`), any collaboration agreement for the `Query-Price` conversation definition in which `XOCP-hub-dc` is assigned the role supplier are identified. One collaboration agreement, `Query-Price IntCo-to-Buyer`, meets the criteria. Based on that agreement, the messages are 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 described in “XOCP Peer-to-Peer Messaging” on page 2-7, where two trading partners, ABC International, a computer manufacturer, and XYZ Systems, a chip supplier, plan to use WebLogic Collaborate to participate in Query Price and Availability (QPA) transactions. In this case, the trading partners will be using the RosettaNet protocol and employing PIP 3A2 to carry out the public processes.

Trading partners participating in 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 Collaborate distribution has been customized and connected to private processes that interact with internal systems as required. For the purposes of this example, the collaborative workflows are named `PIP3A2_Supplier` and `PIP3A2__Buyer`.

For general information on using the WebLogic Collaborate plug-in with WebLogic Process Integrator to create collaborative workflows, see *Creating Workflows for BEA WebLogic Collaborate*. For information specific customizing the PIP templates provided with WebLogic Collaborate and configuring RosettaNet security, see *Implementing RosettaNet for BEA WebLogic Collaborate*.

The sections that follow provide an example of how to:

- Configure the environment to support RosettaNet messaging.
- 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 in collaboration agreements

The Environment

Before you can support RosettaNet messaging in a domain, you must do the following:

- Copy the Document Type Definitions (DTDs) from the `collaborate/rosettanet/dtds` directory into the WebLogic Server runtime directory.

  The WebLogic Server runtime directory is the parent directory of the `confg/`*`domain_name`* domain directory. For example, for the custom domain, `<`*`BEA_Home`*`>/wlintegration2.0/collaborate/config/mydomain`, the WebLogic Server runtime directory is `<`*`BEA_Home`*`>/wlintegration2.0/collaborate`. Therefore, to support RosettaNet messaging in the custom domain, copy the DTDs from `<`*`BEA_Home`*`>/wlintegration2.0/collaborate/rosettanet/dtds` to `<`*`BEA_Home`*`>/wlintegration2.0/collaborate`.

- Copy the `collaborate/rosettanet/schemas` directory, and its contents, under the WebLogic Server runtime directory.

  For example, for the custom domain, `<BEA_Home>/wlintegration2.0/collaborate/config/mydomain`, the WebLogic Server runtime directory is `<BEA_Home>/wlintegration2.0/collaborate`. Therefore, to support RosettaNet messaging in the custom domain, copy the `<BEA_Home>/wlintegration2.0/collaborate/rosettanet/schemas` directory to `<BEA_Home>/wlintegration2.0/collaborate`.

When you have made the required changes, the DTDs should be in the *`weblogic_server_runtime`* directory, and the structure for the domain should appear as shown in the following figure.

Figure 2-20 RosettaNet Directories

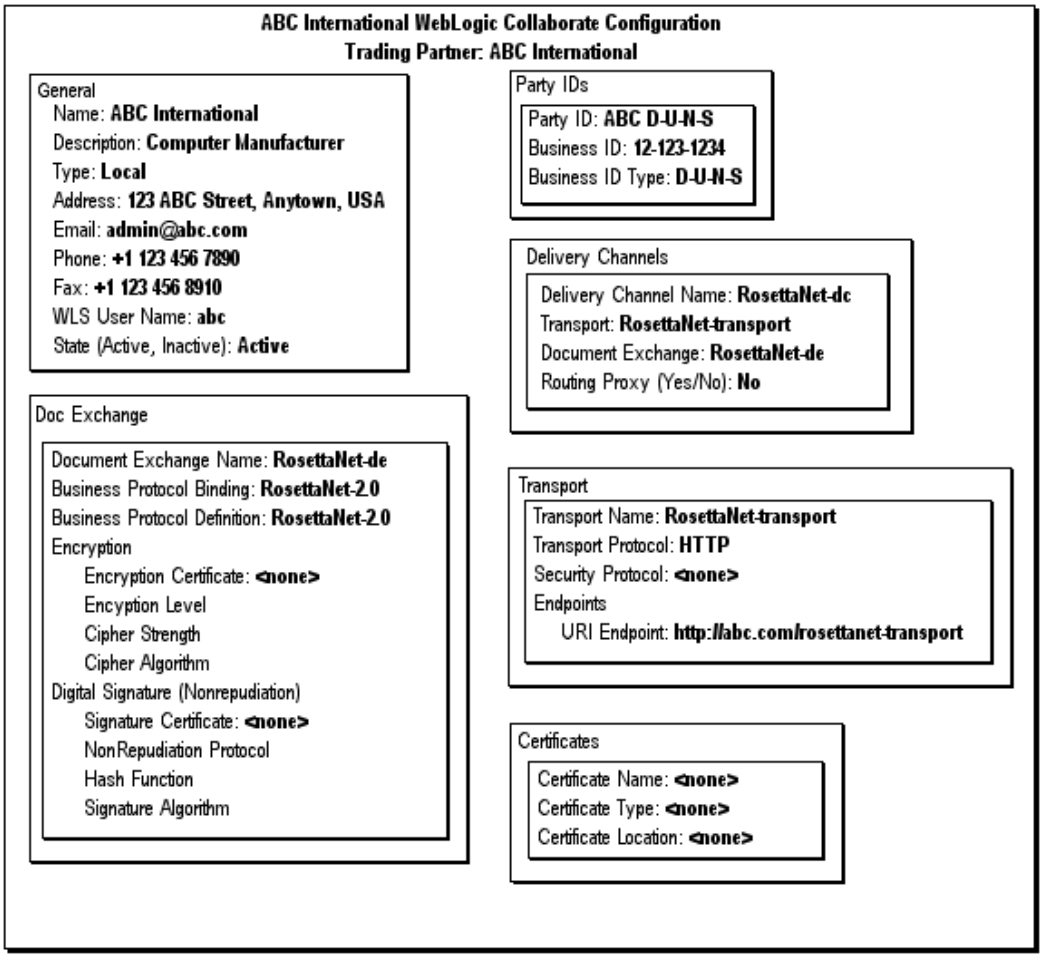
Trading Partners

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

Note: For simplicity, SSL or signature certificates are not used in the examples presented in this section. For information about the security configuration requirements, see *Using BEA WebLogic Collaborate Security*.

The requirements for the ABC International trading partner in the ABC configuration are summarized in the following figure.

Figure 2-21 ABC International Trading Partner Definition in the ABC Configuration

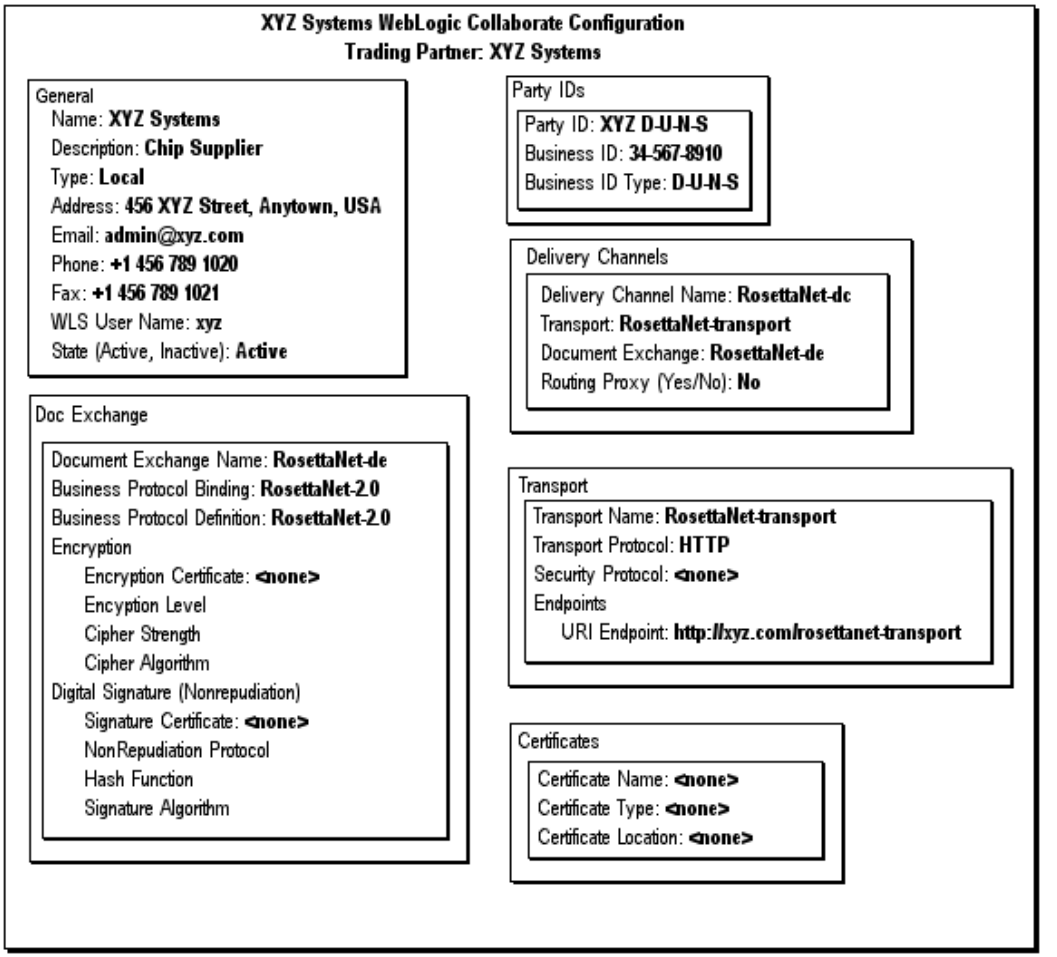


Note that the trading partner type is `Local` and the configuration includes a single RosettaNet delivery channel (`RosettaNet-dc`), document exchange (`RosettaNet-de`), and transport (`RosettaNet-transport`). The transport has an associated URI which serves as the delivery channel endpoint (`http://abc.com/rosettanet-transport`).

The requirements for the ABC International trading partner in the XYZ WebLogic Collaborate configuration are exactly the same as those just listed, with the exception of trading partner type. Trading partner type is set to `Remote` in the XYZ WebLogic Collaborate configuration.

The requirements for the XYZ Systems trading partner in the XYZ WebLogic Collaborate configuration are summarized in the following figure.

Figure 2-22 XYZ Systems Trading Partner Definition in the XYZ Configuration



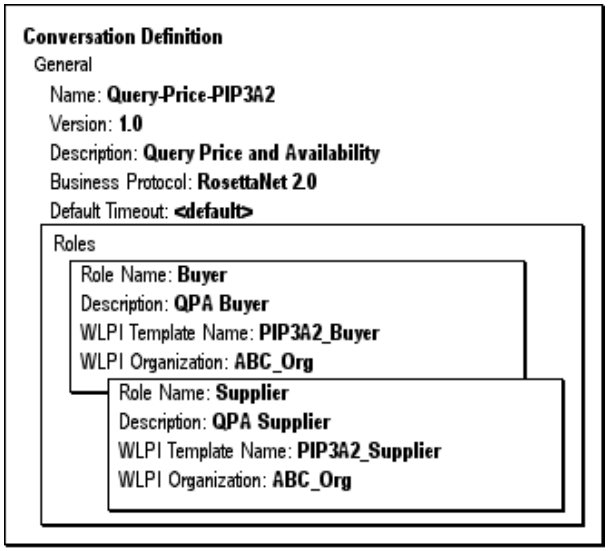
Note that the trading partner type is `Local` and the configuration includes a single RosettaNet delivery channel (`RosettaNet-dc`), document exchange (`RosettaNet-de`), and transport (`RosettaNet-transport`). The transport has an associated URI which serves as the delivery channel endpoint (`http://xyz.com/rosettanet-transport`).

The requirements for the XYZ Systems trading partner in the ABC WebLogic Collaborate configuration are exactly the same as those just listed, with the exception of trading partner type. Trading partner type is set to `Local` in the ABC WebLogic Collaborate configuration.

Conversation Definitions

The requirements for the Query Price and Availability (QPA) conversation definition in the ABC WebLogic Collaborate configuration are summarized in the following figure.

Figure 2-23 Conversation Definition for PIP 3A2



The conversation definition in the XYZ WebLogic Collaborate configuration is the same, with the exception of the WebLogic Process Integrator (WLPI) organization, which reflects the structure used at XYZ Systems.

Collaboration Agreements

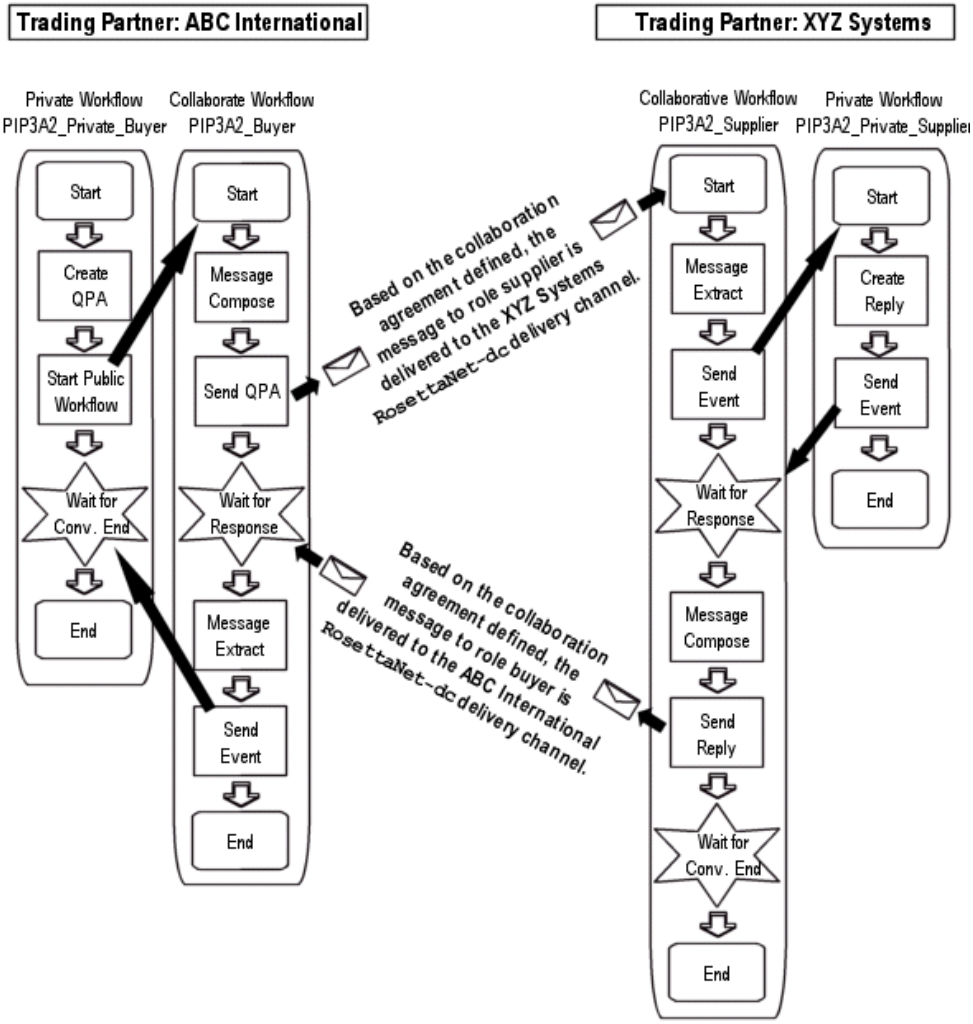
The collaboration agreement shown in the following figure is required in both the ABC and XYZ WebLogic Collaborate configurations.

Figure 2-24 Collaboration Agreement Query-Price-PIP3A2-ABC-XYZ

How It Works

The following figure summarizes how WebLogic Collaborate supports the exchange of Query Price and Availability messages between ABC International and XYZ Systems using RosettaNet 2.0.

Figure 2-25 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 Collaborate, cXML security, and using the cXML API, see *Implementing cXML for BEA WebLogic Collaborate*.

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, constitute the PunchOutSetup transaction.

The structure of each document adheres to the cXML DTD for the 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 2-26 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 2-1 cXML Requirements

For a . . .	This parameter . . .	Must Match . . .
Trading Partner	Business ID Type	The value of the Credential domain attribute. For example if `<Credential domain=”DUNS”>`, then the business ID type must be set to `DUNS`.
	Business ID	The content of the Credential Identity element. For example, if `<Credential domain=”DUNS>` `<Identity>012345678</Identity>`, then the business ID must be set to `012345678`
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 `<Request>` `<OrderRequest> . . .</OrderRequest>`, then the conversation definition name must be set to `Order`.
	Version	The value of the cXML version attribute. For example, if `<cXML version=”1.1.009”. . . .>`, 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.

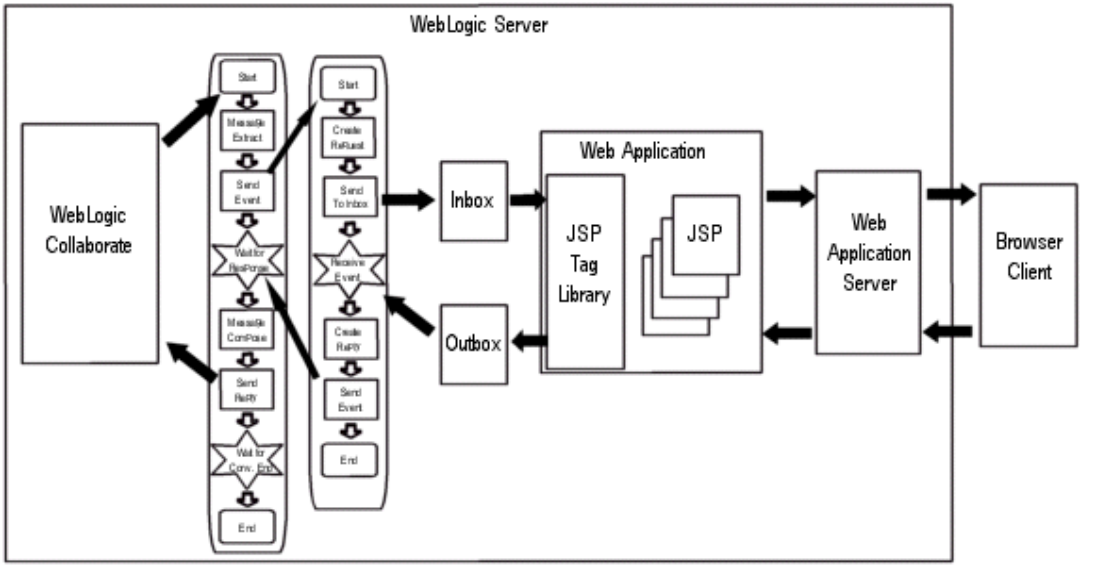
When a trading partner has no requirements for integration with backend systems, hosting the components required to allow that partner to participate as a browser client is the appropriate solution. File-sharing clients often have some requirement for backend systems integration, and usually process 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 end point. This processing must be hosted on behalf of the trading partner.

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

Figure 2-27 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 Collaborate distribution. It provides the interface to mailboxes, supports the creation and removal of mailboxes, and the allows Browser clients to adminster stored messages.

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

- For the incoming mailbox use: *`trading_partner_name`*`_Inbox`
- For outgoing mailbox use: *`trading_partner_name`*`_Outbox`

WebLogic Process Integrator workflows provide the interface between the mailboxes and WebLogic Collaborate. 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.

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 mail box. 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 Collaborate 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 *Using BEA WebLogic Collaborate Samples*.

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

The following example summarizes the configuration required to support a browser client.

Hosting a Browser Client

In this example, the situation is the same as described in "RosettaNet Applications" on page 2-27, where 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 browser client.

The collaborative and private buyer workflows, `PIP3A2_Buyer` and `PIP3A2_Private_Buyer`, are the same as in the RosettaNet example. The collaborative and private workflows, `PIP3A2_Supplier` and `PIP3A2_Private_WebSupplier`, for the supplier role are assumed to have been customized as required 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 2-21

  The trading partner definition for XYZ Systems is the same as the definition shown in Figure 2-22, 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 2-23

- *Collaboration Agreements*
  The collaboration agreement is the same as the agreement shown in Figure 2-24, except that the trading partner name for XYZ Systems is modified (space replaced by underscore)

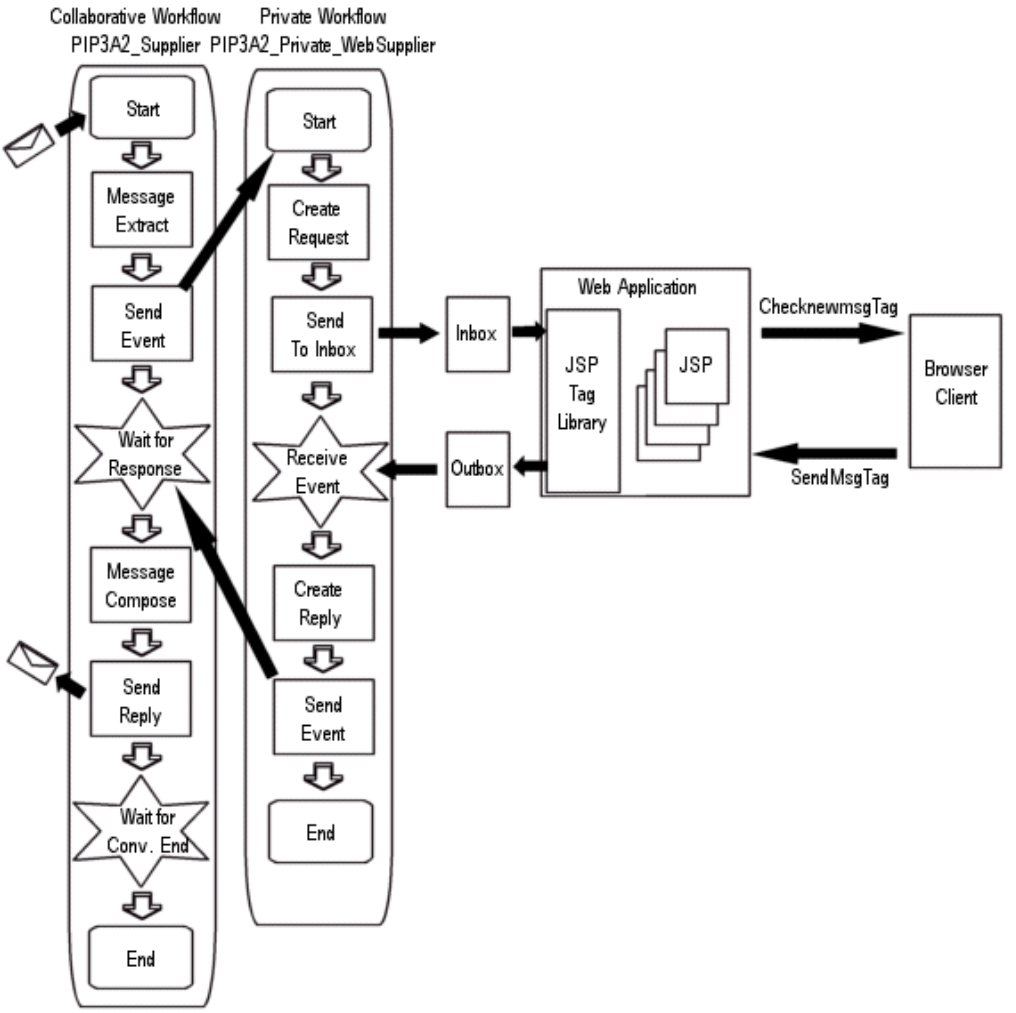
The exchange of messages between the PIP3A2__Buyer and PIP3A2__Supplier workflows is essentially the same as the exchange shown in Figure 2-25. The only difference is that in this example, the exchange takes place within the same WebLogic Collaborate instance.

The processing required to support XYZ Systems as a browser client occurs in the Web application and in the PIP3A2_Private_WebSupplier workflow. This workflow now:

- Transforms the message received from the Send Event task node of PIP3A2_Public_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 an message appropriately formatted for receipt by the Wait for Response event node of the PIP3A2_Public_Supplier

The following figure provides a summary of the actions between the receipt of a message on the Start node and the Send Reply task of the PIP3A2_Private_WebSupplier workflow.

Figure 2-28 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 not integration with backend systems.

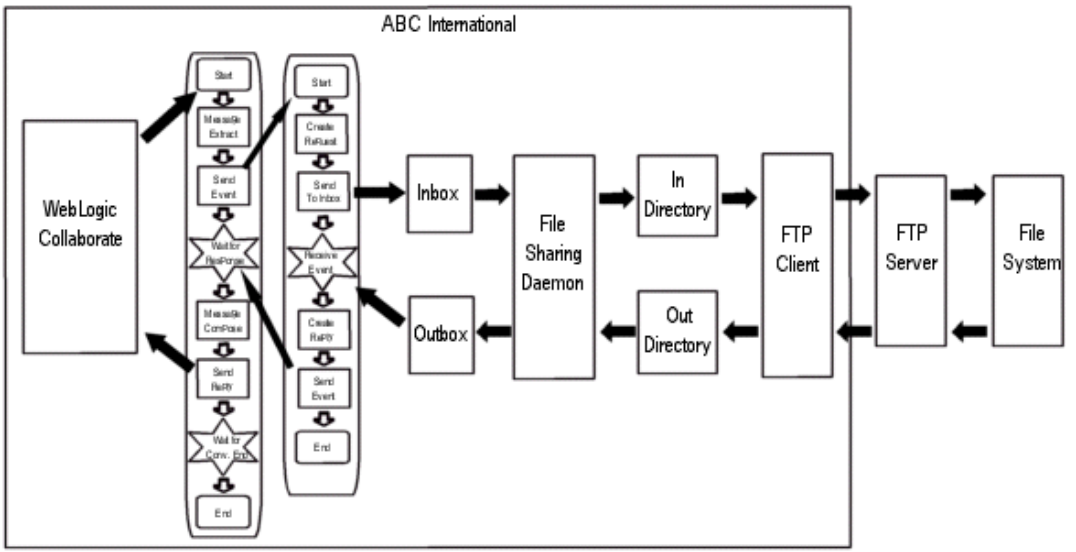
When a trading partner has minimal requirements for integration with backend systems, hosting the components required to allow that partner to participate as a file-sharing client is usually the appropriate solution. File-sharing clients often process a higher volume of messages than browser clients.

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 end point. This processing must be hosted on behalf of the trading partner.

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

Figure 2-29 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 `CreateboxTag`. 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 outgoing mailbox use: *`trading_partner_name`*`_Outbox`

WebLogic Collaborate 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.

A customer or third party supplied FTP client 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 configuration required to support a file-sharing client.

Hosting a File-Sharing Client

In this example, the situation is the same as described in “Browser Clients” on page 2-38, where 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 buyer workflows, `PIP3A2__Buyer` and `PIP3A2__Private_Buyer`, are the same as in the browser client example. The collaborative and private workflows, `PIP3A2_Supplier` and `PIP3A2_Private_FTPSupplier`, for the supplier role are assumed to have been customized as required to support the 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 File-Sharing Client" in "Trading Partner Lightweight Client Sample" of *Using BEA WebLogic Collaborate 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 2-21

  Trading partner definition for XYZ Systems is the same as the definition shown in Figure 2-22, 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 2-23

- *Collaboration Agreements*
  The collaboration agreement is the same as the agreement shown in Figure 2-24, except that the trading partner name for XYZ Systems is modified (space replaced by underscore).

The exchange of messages between the `PIP3A2__Buyer` and `PIP3A2__Supplier` workflows is essentially the same as the exchange shown in Figure 2-25. The only difference is that now, this exchange takes place within the same WebLogic Collaborate instance.

The processing required to support XYZ Systems as a file-sharing client occurs at the file-sharing location and in the `PIP3A2_Private_WebSupplier` workflow. This workflow now:

- Transforms the message received from Send Event task node of `PIP3A2_Public_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 an message appropriately formatted for receipt by the Wait for Response event node of the `PIP3A2_Public_Supplier`

3 Basic Configuration Tasks

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

- WebLogic Collaborate Administration Console Overview
- Configuring WebLogic Collaborate
- 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 WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

For examples and discussion of the configuration requirements for selected scenarios, see Chapter 2, "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 XOCP business messages, are discussed in Chapter 4, "Advanced Configuration Tasks."

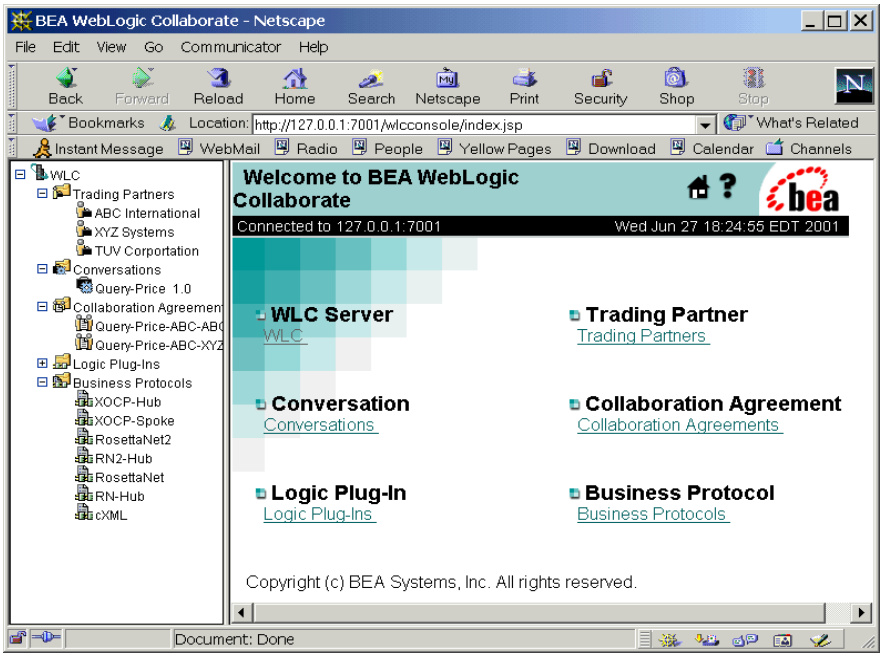
WebLogic Collaborate Administration Console Overview

The WebLogic Collaborate Administration Console is used to:

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

When you start the WebLogic Collaborate Administration Console, as described in "Starting the WebLogic Collaborate Administration Console" on page 1-7, the following page is displayed.

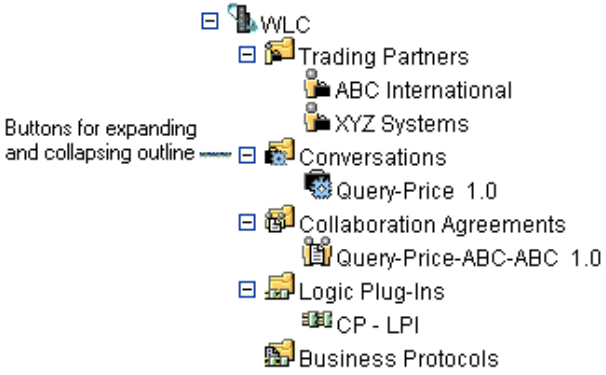
Figure 3-1 WebLogic Collaborate Administration Console



Like the WebLogic Server Administration Console, the navigation tree is used to navigate to the appropriate page. If the Display entities on the navigation tree option is checked on the WebLogic Collaborate Preferences tab, the trading partners, conversation definitions, and collaboration agreements defined for your application are also available for selection from the navigation tree, as shown in the following figure.

Note: For information about the Display entities on the navigation tree option, see "Displaying Items in the Navigation Tree" in *BEA WebLogic Collaborate Administration Console Online Help*.

Figure 3-2 Navigation Tree



The navigation methods for the WebLogic Collaborate Administration Console are summarized in the following figures.

Note: The controls for the logic plug-ins and business protocols are not shown in the figures. If the Hide Advanced Controls option is checked on the WebLogic Collaborate Preferences tab, the Logic Plug-Ins tab is not displayed and the Business Protocols tab cannot be selected (that is, you cannot create or configure logic plug-ins, or view or modify the default business protocol definitions). These advanced features are discussed in Chapter 4, "Advanced Configuration Tasks."

Figure 3-3 WebLogic Collaborate Administration Console Navigation

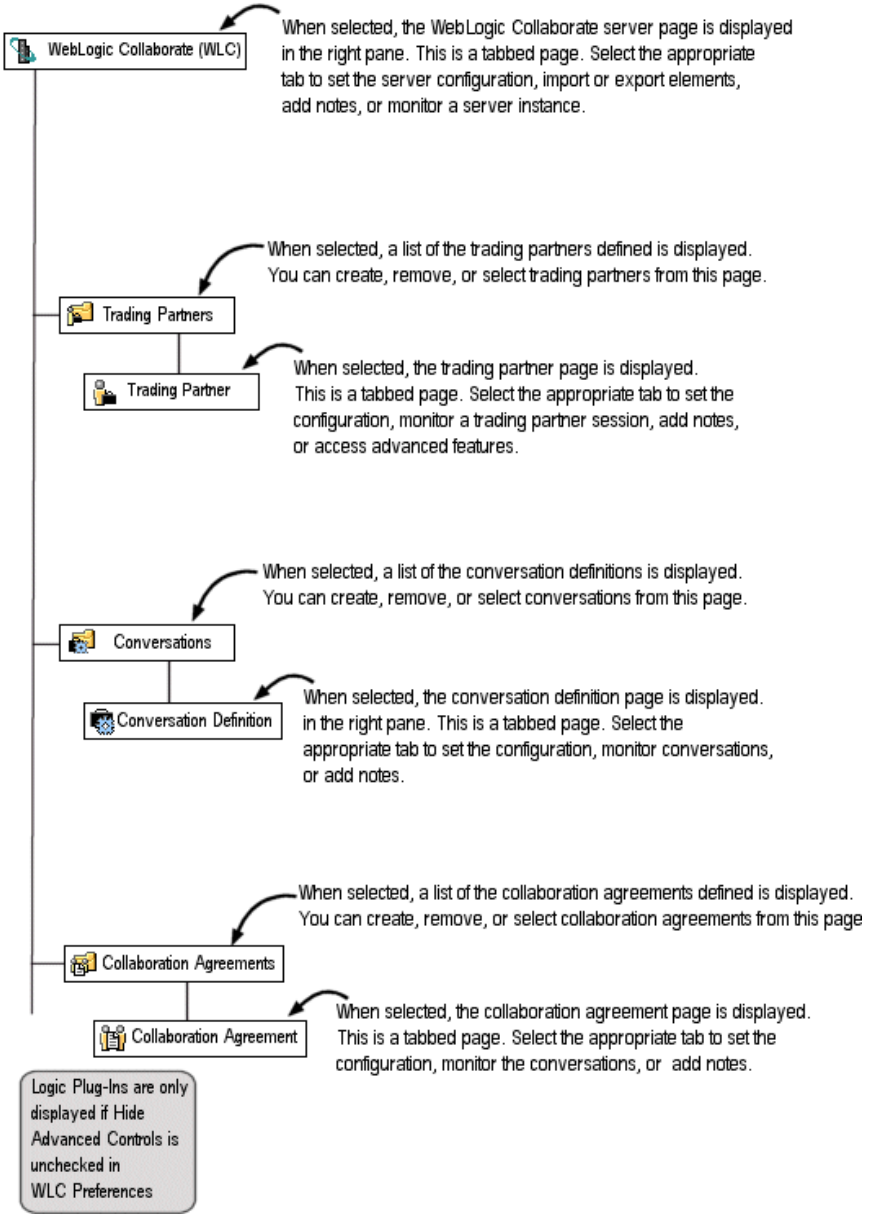
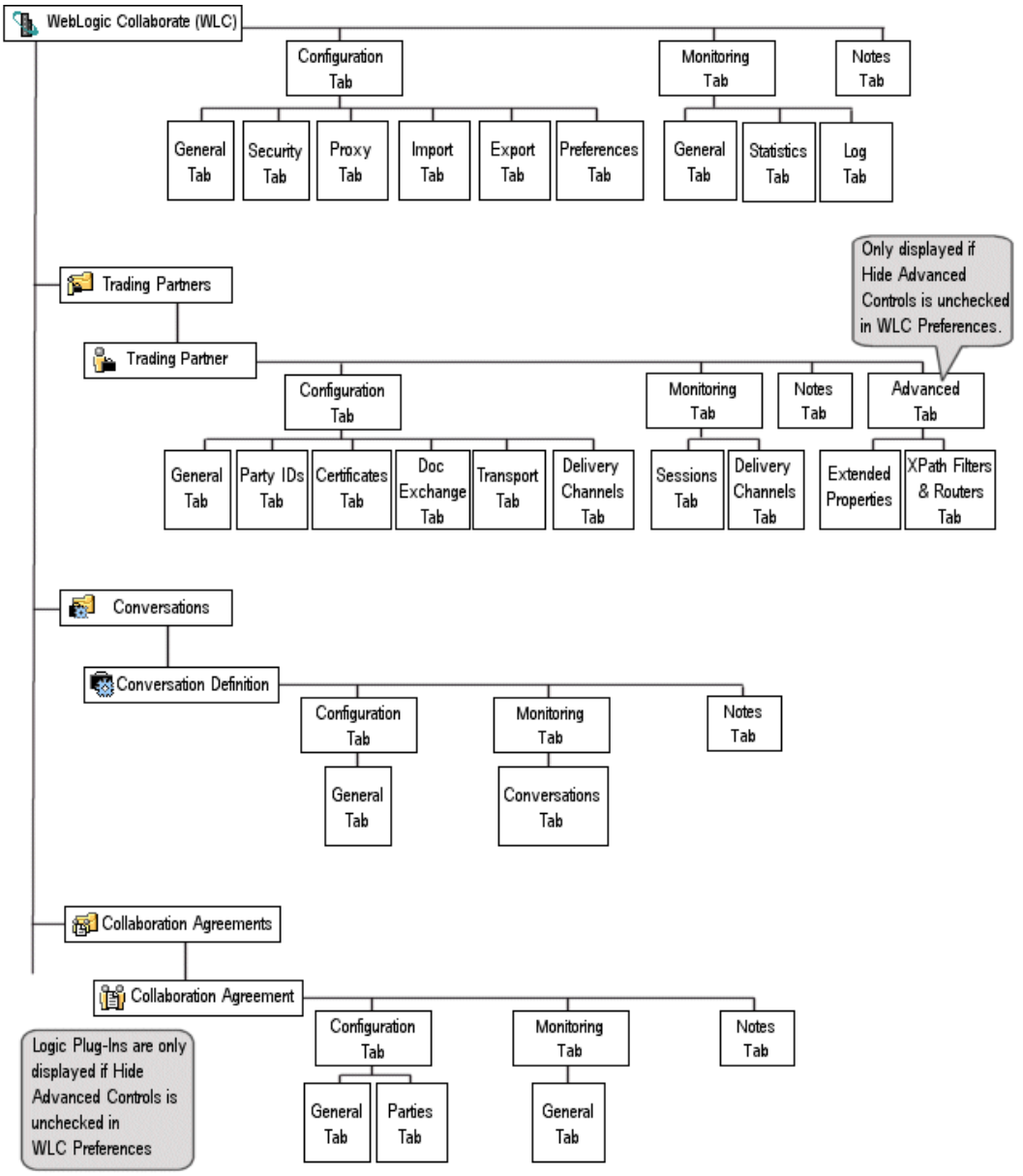


Figure 3-4 WebLogic Collaborate Administration Console Navigation



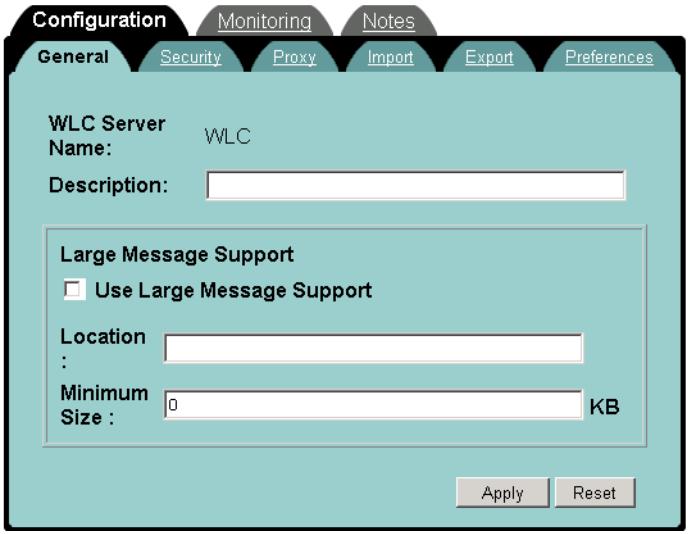
Configuring WebLogic Collaborate

When you select WebLogic Collaborate (WLC) from the navigation tree, the WebLogic Collaborate page is displayed. This is a tabbed page from which you can:

- View or modify the WebLogic Collaborate configuration
- Monitor the WebLogic Collaborate instance
- Add notes to the configuration
- Import or Export configured trading partners, conversation definitions, or collaboration agreements

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

Figure 3-5 Configuring WebLogic Collaborate



Configuring the WebLogic Collaborate server involves configuring the following items:

- WebLogic Collaborate parameters
- Security
- Proxy
- Preferences

Tasks related to configuration that are performed from the WebLogic Collaborate page include:

- Exporting repository data
- Importing repository data

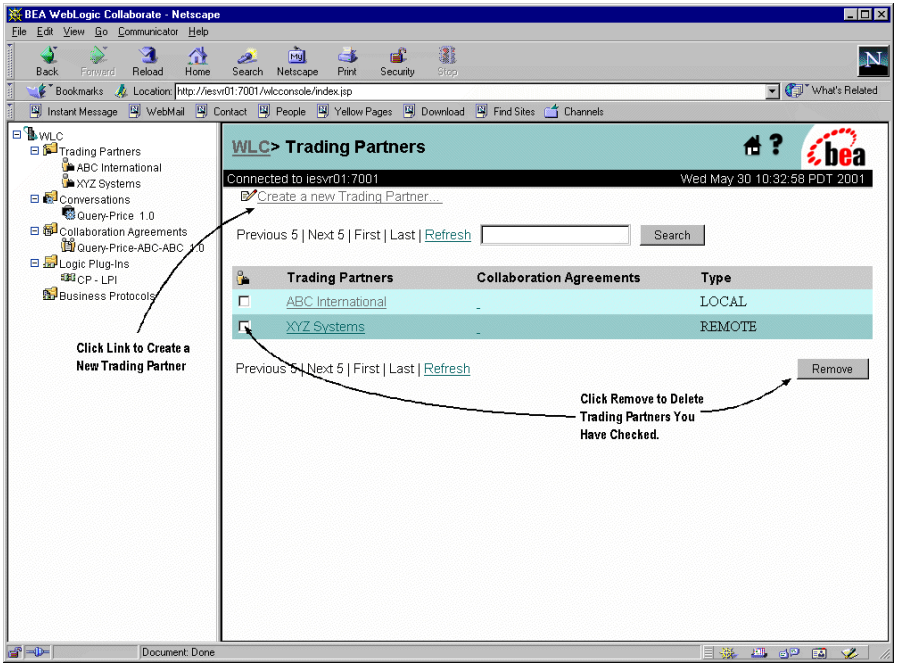
The detailed information required to perform these tasks is provided in the WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

Configuring Trading Partners

When you select Trading Partners from the navigation tree, a list of the currently defined trading partners is displayed. As shown in the following figure, you can select or search for an existing trading partner, create a new trading partner, or remove an existing trading partner. Clicking on a trading partner in this list is equivalent to selecting the trading partner from the navigation tree.

Figure 3-6 Trading Partners Page



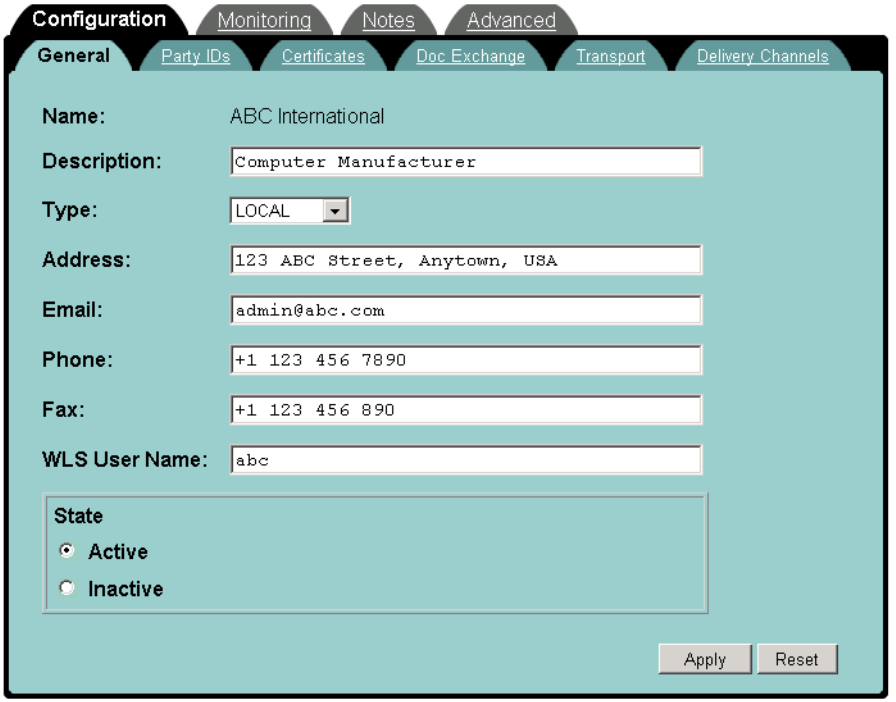
When you select Create a New Trading Partner on the Trading Partners page, or select an existing trading partner from the navigation tree, the trading partner page is displayed. This is a tabbed page from which you can:

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

- Add notes to the trading partner configuration
- 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

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

Figure 3-7 Trading Partner Page



Configuring a trading partner involves configuring 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 to perform these tasks is provided in the WebLogic Collaborate Administration Console online help.

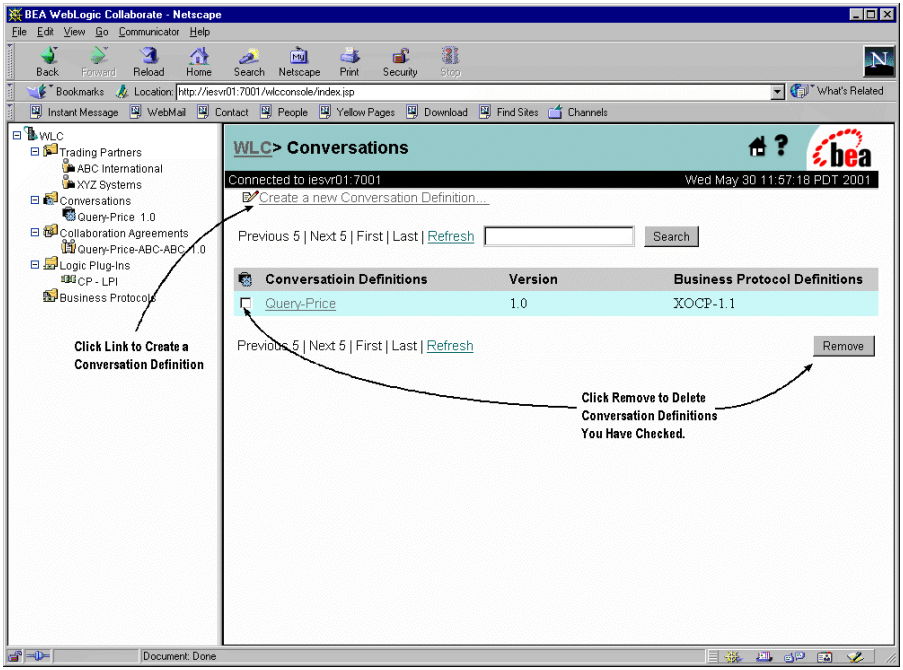
Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

In addition to the preceding tasks, you can also configure extended properties for a trading partner, and XPath router and filter expressions. For a discussion of these advanced features, refer to Chapter 4, "Advanced Configuration Tasks."

Configuring Conversation Definitions

When you select Conversations from the navigation tree, a list of conversation definitions is displayed. As shown in the following figure, you can select or search for an existing conversation definition, create a new conversation definition, or remove an existing one. Clicking on a conversation definition in this list is equivalent to selecting the conversation definition from the navigation tree.

Figure 3-8 Conversations Page

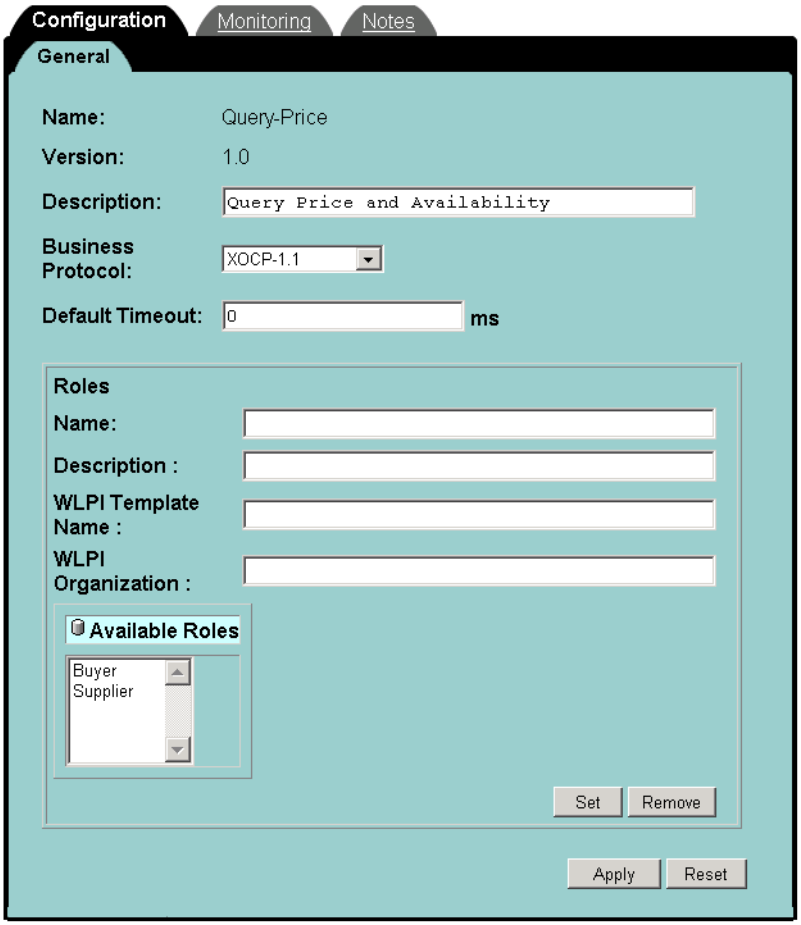


When you select Create a New Conversation Definition on the Conversations page, or select an existing conversation definition from the navigation tree, the conversation definition page is displayed. This is a tabbed page from which you can:

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

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

Figure 3-9 Conversation Definition Page



Configuring a conversation definition involves the following tasks:

- Configuring basic identifying information
- Defining the conversation roles
- Associating roles with a WebLogic Process Integrator workflow template and organization

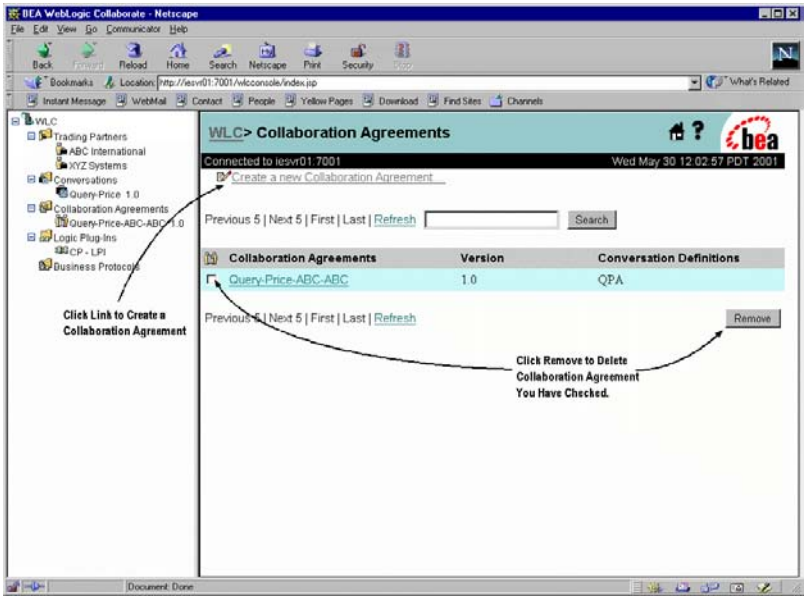
The detailed information required to perform these tasks is provided in the WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

Configuring Collaboration Agreements

When you select Collaboration Agreements from the navigation tree, a list of the currently defined collaboration agreements is displayed. As shown in the following figure, you can select or search for an existing collaboration agreement, create a new collaboration agreement, or remove an existing one. Clicking on a collaboration agreement in this list is equivalent to selecting the collaboration agreement from the navigation tree.

Figure 3-10 Collaboration Agreement Page

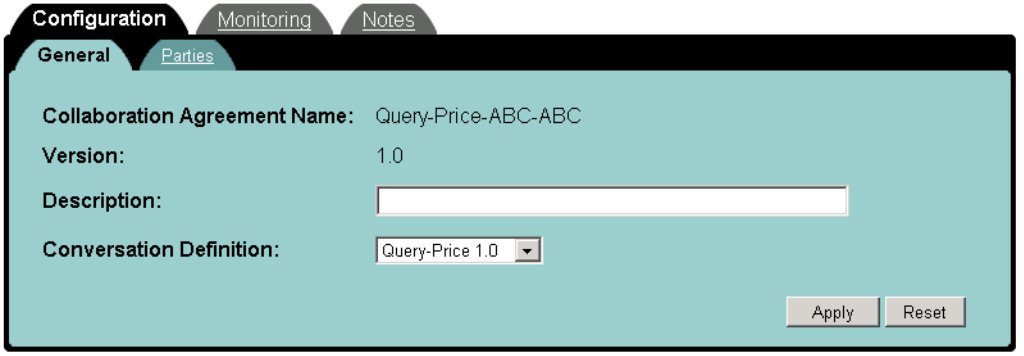


When you select Create a New Collaboration Agreement on the Collaboration Agreements page, or select an existing collaboration agreement from the navigation tree, the collaboration agreement page is displayed. This is a tabbed page from which you can:

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

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

Figure 3-11 Collaboration Agreement Page



Configuring a collaboration agreement involves defining the following:

- Basic identifying information
- Collaboration agreement parties

The detailed information required to perform these tasks is provided in the WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

CHAPTER

4 Advanced Configuration Tasks

This section serves as an introduction to the more advanced features of WebLogic Collaborate and provides a roadmap to where more detailed information can be found. It includes the following topics:

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

Overview of Advanced Features

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

- *Routing and filtering business messages*
  The built-in XOCP router and XOCP filter logic plug-ins support customer-defined XPATH router and filter expressions. Based on the XPath router and filter expression defined, you can control the flow of XOCP business messages exchanged among trading partners as follows:
  - The XPath router logic plug-in can modify the list of recipients for an XOCP business message.
  - The XPath filter logic plug-in can determine whether an XOCP 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 XOCP business protocol.
- *Custom logic plug-ins*
  A default chain of built-in, business-protocol specific, logic plug-ins are associated with each business protocol routing and filtering functions. 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 (the services are not limited to routing and filtering).

In addition, WebLogic Collaborate supports the specification of trading partner extended properties. 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.

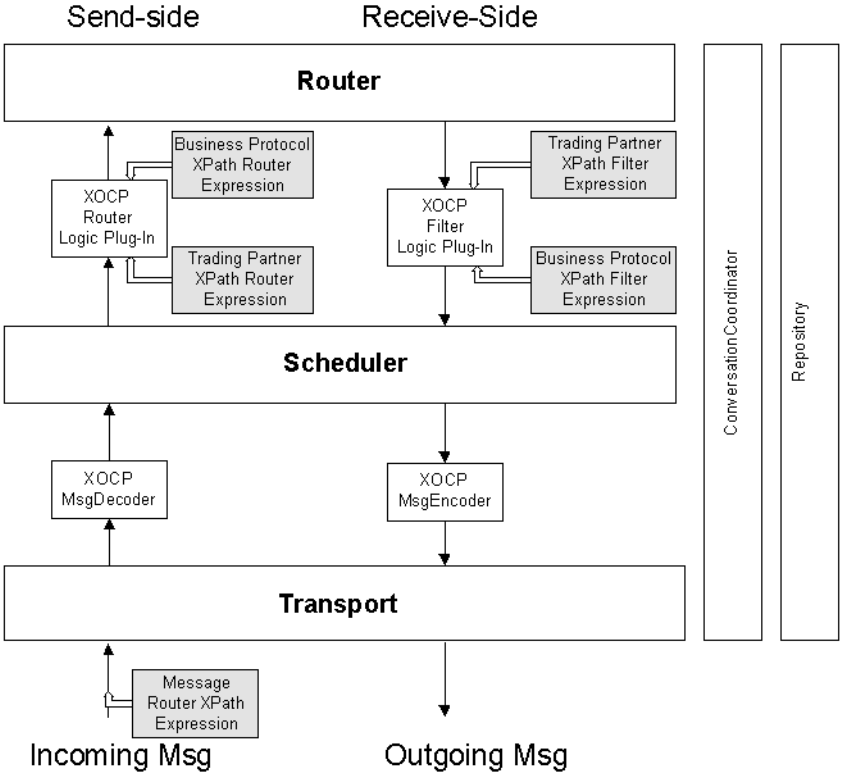
- *XPath Expressions added when the business message is assembled*
  Message XPath router expressions can be associated directly with a message. When an XOCP message is assembled, the trading partner recipient can be left blank, identified by name, or identified by XPATH expression. For example, if you are using the WebLogic Collaborate plug-in to create a collaborative workflow in WebLogic Process Integrator Studio, you can 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 BEA WebLogic Collaborate*.)
- *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 WebLogic Collaborate Administration 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 WebLogic Collaborate Administration 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 2-5, 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 trading partner sender and recipient(s) as identified by the conversation definition and collaboration protocol agreements configured. Any XPath router expressions defined use the XPath syntax to select a set of trading partners from the 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 the message processing and routing on an XOCP hub delivery channel.

Figure 4-1 XOCP Hub Delivery Channel Message Processing



As noted, when an incoming message is received by a hub delivery channel, the conversation definition and associated collaboration agreements 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) which defines the roles buyer and supplier. Trading partner 1 (TP1) has a delivery channel (`tp1-hub-dc`) defined, while trading partners 2, 3, 4, and 5 (TP2, TP3, TP4, TP5) each have a spoke delivery channel 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`) assigned to QPA role supplier
  TP2 (`tp2-spoke-dc`) assigned to QPA role buyer

- TP1 (`tp1-hub-dc`) assigned to QPA role buyer
  TP3 (`tp3-spoke-dc`) assigned to QPA role supplier
  TP4 (`tp4-spoke-dc`) assigned to QPA role supplier
  TP5 (`tp5-spoke-dc`) assigned to QPA role supplier

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

In this case, the XML message-context document generated by the XOCP router logic plug-in would contain the identifying information and properties for the sender, TP1, and the recipients, TP3, TP4, and TP5.

Any XPath router expressions defined would be used by the XOCP 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 append to the results of the previous expression. For additional information about how the XPath router expressions are processed, see "XPath Router Expression Processing" on page 4-6.

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`), nothing else about the document is changed. In other words, all XPath router expressions are used to select from the set of trading partners originally included.

Once the XOCP router logic plug-in has completed processing and messages are routed to the trading partners identified, the XPath filter logic plug-in generates an XML message-context document for each outgoing business message. If XPath filter expressions have been defined, these are evaluated against the message-context document generated by the XOCP filter to determine whether to send the business message 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 further 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 4-7.

As noted in "Overview of Advanced Features" on page 4-2, the XPath router and filter expressions can reference user-defined extended properties. (For a discussion of trading partner extended properties, see "Trading Partner Extended Properties" on page 4-14.)

XPath Router Expression Processing

As described in the preceding section, XPath router expressions fall into three categories. The expressions are processed in the following order. Each expression is configured to replace or append to the results of the previous XPath expression.

1. *Message XPath router expressions*
   Message XPath router expressions are included in the business message and so 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 and 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 that 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. Within the same type, expressions are processed in the order listed in the WebLogic Collaborate Administration Console.

XPath Filter Expression Processing

As described in the preceding section, XPath filter expressions fall into two categories. The expressions are evaluated in the following order.

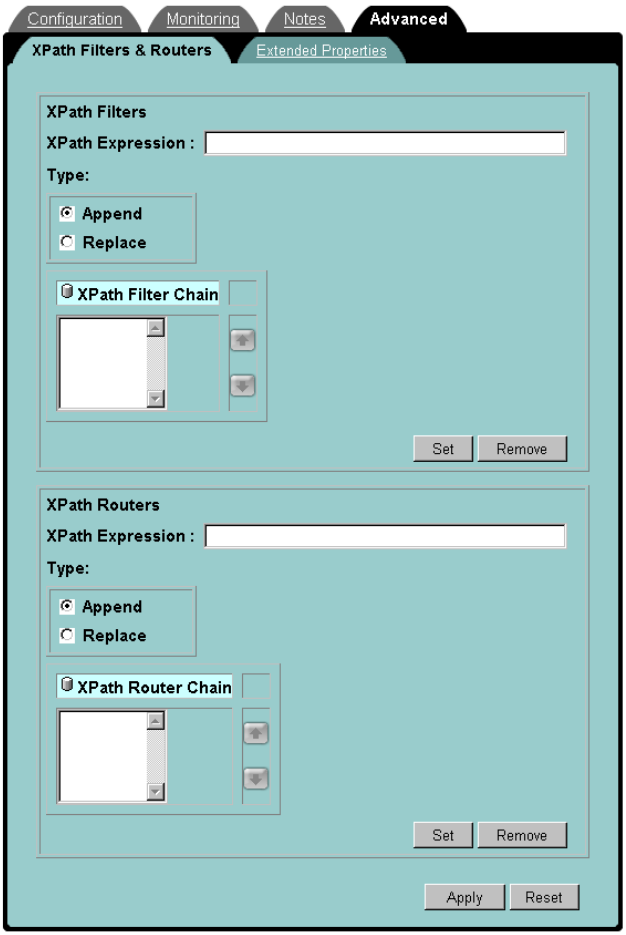
1. *Trading Partner XPath filter expressions*
   Trading partner XPath filter expressions are associated with the recipient trading partner and apply to all messages destined for that trading partner.
2. *Business Protocol XPath filter expressions*
   Business protocol message filter expressions apply to all outgoing business messages using that protocol.

If you define more than one trading partner XPath filter expression or business protocol XPath filter expression, the trading partner XPath filter expressions are evaluated before the business protocol XPath filter expressions. Processing continues until an expression evaluates to false or all expressions have been processed. Within the same type, expressions are processed in the order listed in the WebLogic Collaborate Administration Console.

Configuring Trading Partner and Business Protocol Router and Filter Expressions

When you select the Advanced tab in the trading partner page, then select the Filters & Routers tab, the following is displayed.

Figure 4-2 Trading Partner Filters & Routers



When you select the Configuration tab in the XOCP business protocol definition page, you are presented with an identical Filters & Routers tab (only the context is different).

From the XPath Filters & Routers tab you can configure and order the XPath expressions required. The detailed information required to perform these tasks is provided in the WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

Additional Information

Additional information about routing and filtering business messages, message-context document structure, and creating XPath expressions can be found in "Routing and Filtering Business Messages" in *Programming BEA WebLogic Collaborate Logic Plug-Ins*.

Custom Logic Plug-Ins

As described in "Overview of Advanced Features" on page 4-2, 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 which is used in the processing. By default, this is the first logic plug-in 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 4-3.
- *Router enqueue logic plug-in*
  This logic plug-in adds business messages to the router message queue. By default, this is the last logic plug-in in the router chain.
- *Filter logic plug-in*
  This logic plug-in processes outgoing business messages and generates the message-context document which is used in processing outgoing messages. By default, this is the only logic plug-in in the filter chain. The XOCP filter logic plug-in for a hub delivery channel can use any XPath filter expressions defined to determine whether or not to send a message as described in "XPath Expressions in Routing and Filtering" on page 4-3.

The built-in logic plug-ins are summarized in the following table.

Table 4-1 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 the router or filter processing chain for a business protocol. Although custom logic plug-ins are associated with a router or filter processing chain, they need not perform routing or filtering services. 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 have developed a custom logic plug-in, the logic plug-in definition must be added to a business protocol definition router or filter chain. You can add the logic plug-in from the WebLogic Collaborate Administration Console as follows:

1. Create the logic plug-in definition and specify the following logic plug-in 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 use when initializing the Java class.

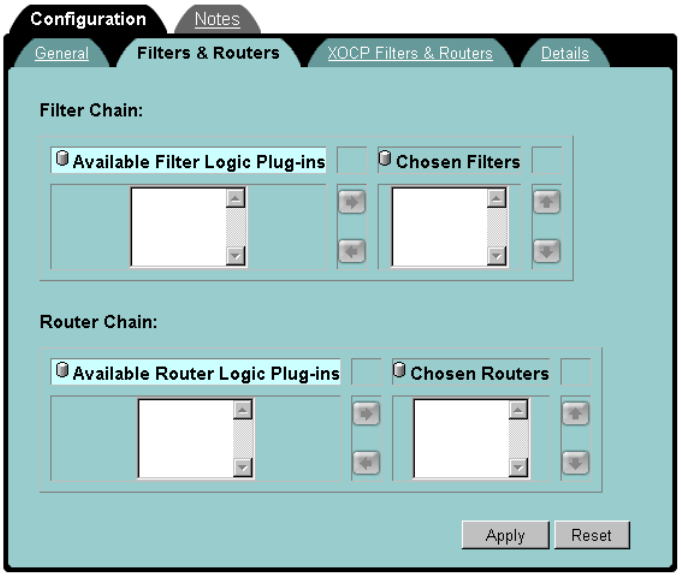
   To create a new logic plug-in definition, select Logic Plug-Ins from the navigation tree, then select Create a New Logic Plug-In. The logic plug-in page is displayed. Fields are provided which allow you to specify the properties specified in the preceding list.
2. Add the logic plug-in to the business protocol definition and specify the position of the logic plug-in in the router or filter chain.

   This step 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 the logic plug-in to a business protocol router or filter processing chain, select the business protocol from the navigation tree, then select the Filters & Routers tab to display the following.

Figure 4-3 Business Protocol Filters & Routers tab



From the Filters & Routers tab you can select and order the available logic plug-ins as required. The detailed information required to perform this task is provided in the WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

Additional Information

Information about the logic plug-in API and guidelines for developing and deploying custom logic plug-ins can be found in *Programming BEA WebLogic Collaborate Logic Plug-Ins*.

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 WebLogic Collaborate Administration Console. Once added, these properties are included in the message-context documents generated by the business protocol router and filter logic plug-ins as uniquely named extended propertys ets.

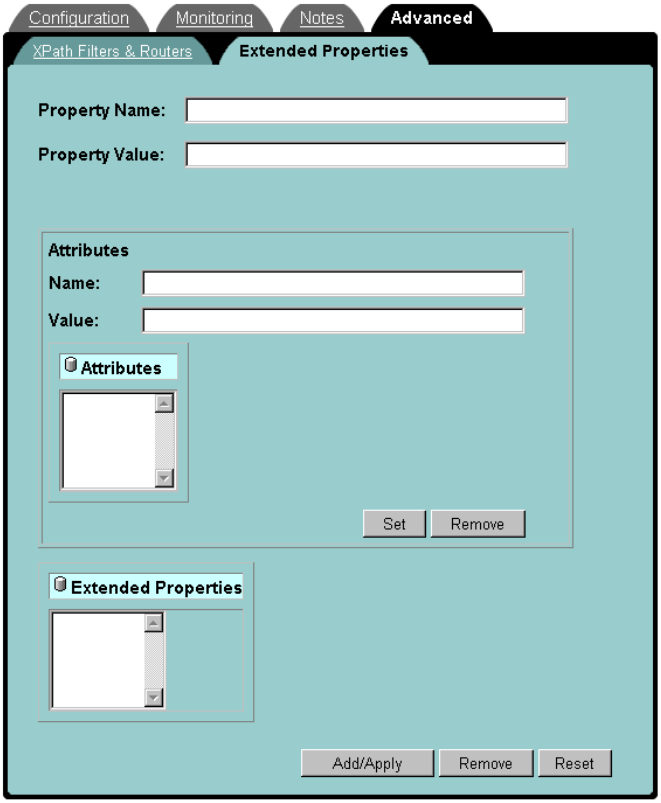
The extended property sets are modeled in the repository so that they can be retrieved as sub-trees within an XML document. These XML sub-trees appear in the generated message-context XML document generated by the built in router and filter logic-plug-ins. The XPath expressions can reference these extended properties. The root elements of each extended property set associated with a given trading partner are be 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, select the trading partner from the navigation tree, select the Advanced tab, then select the Extended Properties tab to display the following.

Figure 4-4 Trading Partner Extended Properties



From the Extended Properties tab you can set the extended properties required. The detailed information required to perform this task is provided in the WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

Additional Information

Additional information about message-context document structure and creating XPath expressions to reference extended properties can be found in *Programming BEA WebLogic Collaborate Logic Plug-Ins*.

CHAPTER

5 Importing and Exporting Business Collaborations

You can facilitate setup for trading partners by creating what they require, then transferring it to them. This section describes how you can export and import the necessary components of a business collaboration. It includes the following topics:

- Business Collaboration Components
- Export and Import Overview
- Exporting from WebLogic Collaborate
- Importing to WebLogic Collaborate
- Exporting a Workflow Package
- Importing a Workflow Package

Business Collaboration Components

The components of a business collaboration 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)

Many of the same components of a business collaboration are used by multiple trading partners. For example, all parties who are assigned the same role in a collaboration agreement for a given conversation definition will require the same conversation definition and public workflow template. To support peer-to-peer exchange between trading partners, both trading partners must have trading partner definitions for themselves and the other party.

To facilitate trading partner setup, the components of a business collaboration can be defined by one partner, then exported for import by other trading partners. Trading partner definitions, collaboration agreements, and conversation definitions can be exported and imported from the WebLogic Collaborate Administration Console. Collaborate workflows can be exported and imported from the WebLogic Process Integrator Studio.

Although many components of a business collaboration are identical between trading partners, there are some that require minor modification after import. For example, if a trading partner definition for a remote trading partner is exported then imported by the remote trading partner, that trading partner will need to change the trading partner type property from remote to local before the imported definition can be used. Workflow templates and conversations too often require modification of the specified organization because different trading partners often have a different organization structure in the WebLogic Process Integrator Studio.

Export and Import Overview

To export the required components for a WebLogic Collaborate exchange:

1. Export the required trading partner definitions, conversation definitions, and collaboration agreements from the WebLogic Collaborate Administration Console.

   Typically, you can simply export the required collaboration agreement with the Export all referenced entities option set. This will export the collaboration agreement, associated conversation definition, and trading partners.

   Note: When you use the Export all referenced entities option, only the referenced objects are exported. For example, even though a trading partner definition may have many delivery channels associated with it, 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 are exported (only the required party identifiers, delivery channels, document exchanges, transports, and certificates).

   The export procedure is described in "Exporting from WebLogic Collaborate" on page 5-4.

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

   The workflow export procedure is described in "Exporting a Workflow Package" on page 5-9.

To import and modify the exported components for use:

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

   One or more XML files may be provided. The import procedure is described in "Importing to WebLogic Collaborate" on page 5-8.

2. Verify the imported components by viewing in the WebLogic Collaborate Administration 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 WebLogic Process Integrator Studio on your system will not be the same as those defined for other trading partners. The organization specified should reflect the organization you will designate for the workflow.

3. Import the collaborative workflow package required to implement your role in the conversation.

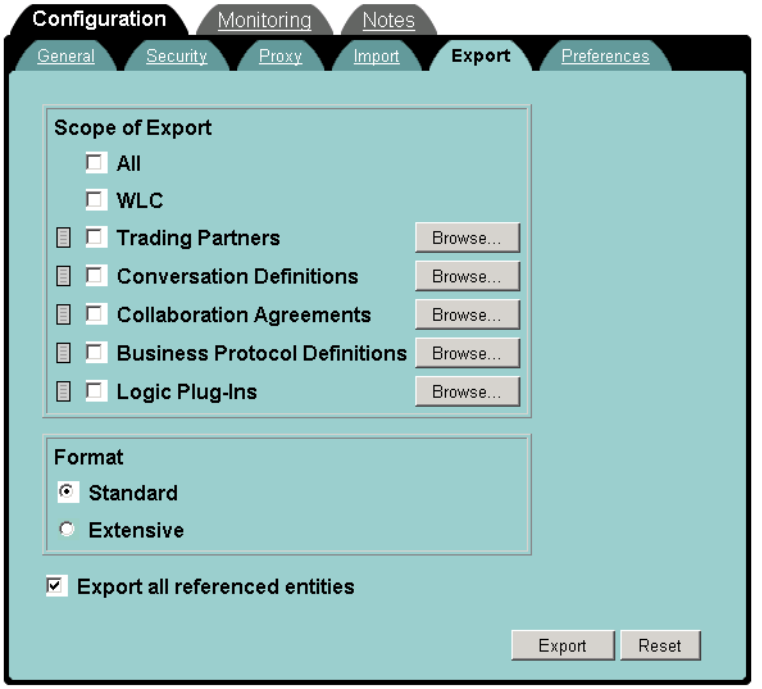
   The workflow import procedure is described in "Importing a Workflow Package" on page 5-11.

Exporting from WebLogic Collaborate

To export entities from the WebLogic Collaborate Administration Console:

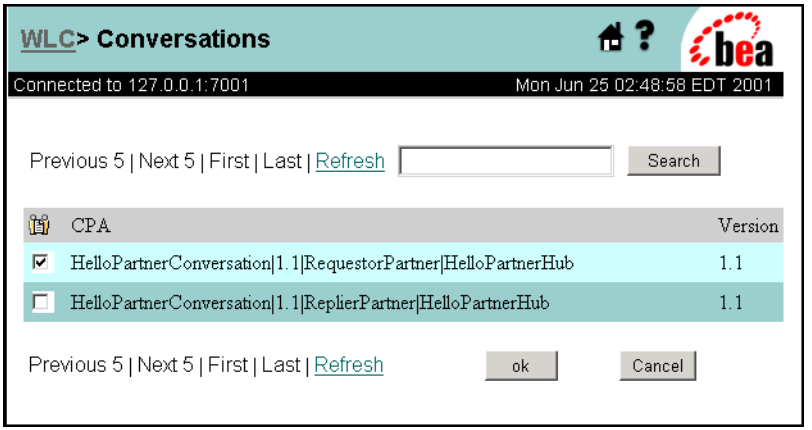
1. Click WebLogic Collaborate (WLC) in the navigation tree to display the WLC page.
2. If it is not already selected, click the Configuration tab.
3. Click the Export tab. The export options are displayed as shown in the following figure.

Figure 5-1 WebLogic Collaborate (WLC) Export Tab



4. Do one or more of the following to select the entities to export:
   - Click the All checkbox to export all repository data.
   - Click the checkbox to the left of the entity label to export all entities of that type.
   - Click the Browse button to the right of the entity label to display the entity selection page as shown following figure. Click the checkboxes as desired to select specific entities, then click the OK button to return to the Export tab.

Figure 5-2 Entity Selection



5. Select the Format.

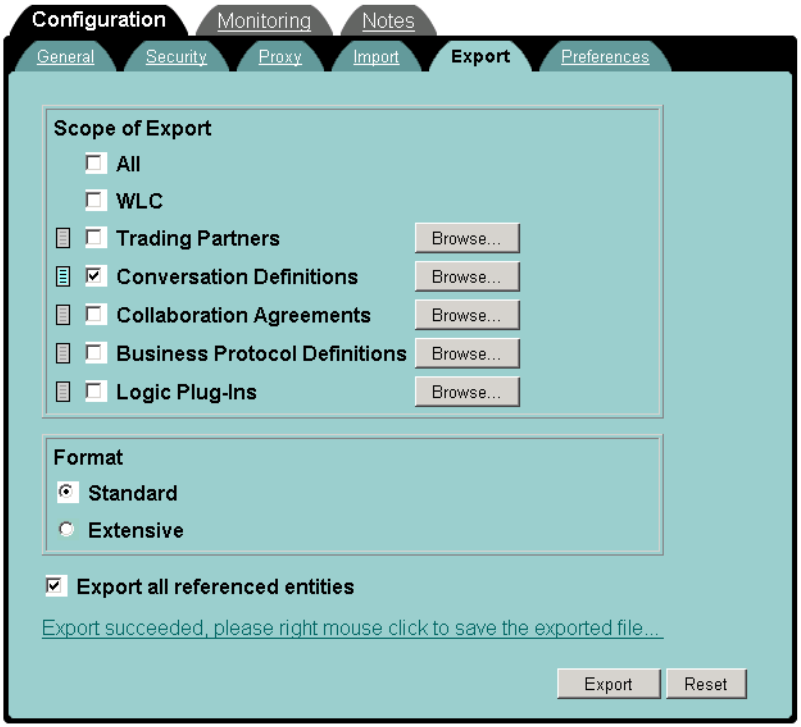
   The default is standard. The extensive format includes system info such as update-count and timestamps.

6. Click the Export all referenced entities to include all entities referenced by the selected entities.

7. Click the Export button to export 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 5-3 Export Succeeded



8. Right-click the link, then select Save Link As from the shortcut menu.

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

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

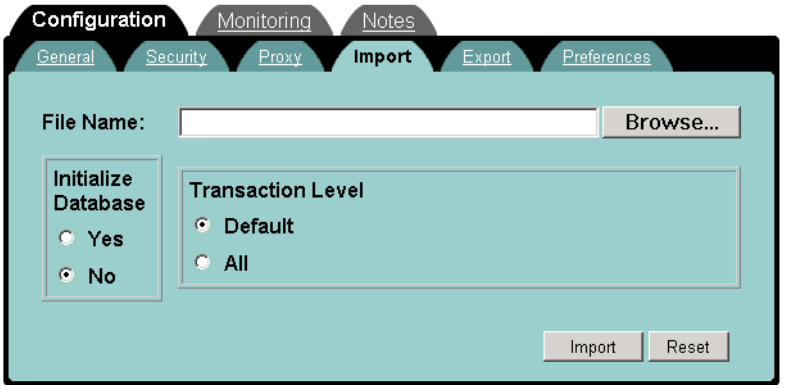
Importing to WebLogic Collaborate

Before importing data, you must shutdown WebLogic Collaborate as described in "Stopping from the WebLogic Collaborate Administration Console" on page 1-13.

To import from the WebLogic Collaborate Administration Console:

1. Click WebLogic Collaborate (WLC) in the navigation tree to display the WLC page.
2. Click the Monitoring tab, then click the General tab.
3. Verify that the Status is Inactive.
4. Click the Configuration tab, then click the Import tab. The import options are displayed as shown in the following figure.

Figure 5-4 WebLogic Collaborate (WLC) Import Tab



5. Click Browse to display the Upload File dialog box.
6. Select the XML file that contains the data to be imported, then click the Open button.
7. Unless it is your intention to delete all existing data, verify that Initialize Database is set to No.

8. Select the Transaction level.
   - *Default*
     A transaction is initiated for each entity type. Entities of the same type are imported in a single transaction. If invalid data is detected during any one of the transactions, that transaction is rolled back and processing continues for the next entity type.
   - *All*
     The data contained in the selected file is imported as a single transaction. If invalid data is detected the entire transaction is rolled back.
9. Click Import to import the data.

   Note: If any entities have the same name as entities in the repository, they will be overwritten.
10. Restart WebLogic Collaborate as described in "Restarting from the WebLogic Collaborate Administration Console" on page 1-15.

Exporting a Workflow Package

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

When you export a workflow package, you can:

- Password protect the package to prevent unauthorized access to the contents.
- Publish the package to make the components read only. When you publish a package, trading partners can employ the workflow components, but are unable to modify them.

The Studio client 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 Studio. When you export a template definition, it is disassociated from the current organization. When imported to a target system, a new organization must be selected for the template.

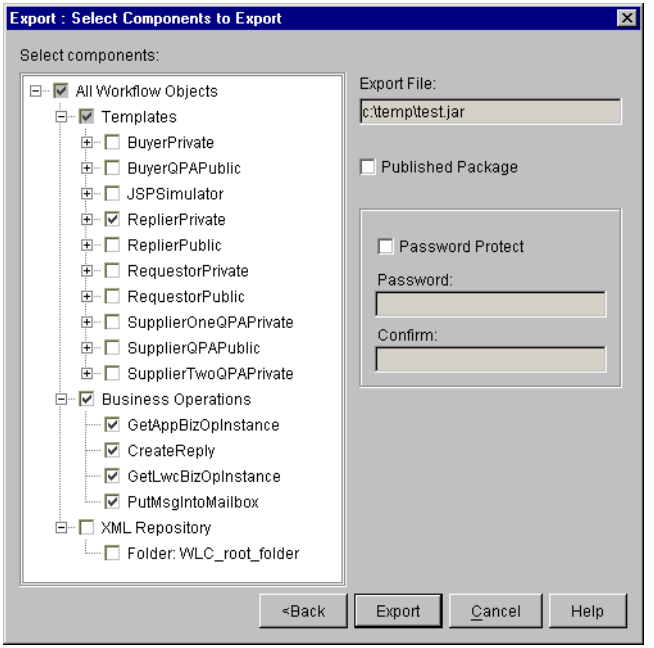
To export a workflow package from Studio:

1. Start the Studio client as described in "Starting WebLogic Process Integrator Studio" on page 1-19.
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 use Browse to select a JAR file, the export will overwrite the file.
4. Click Next to display the Export: Select Components to Export dialog box as shown in the following figure.

Figure 5-5 Export: Select Components to Export



The dialog lists all items currently defined. When you select an item, items referenced by the selection are automatically selected.

Note: Items in the XML Repository are not automatically selected. You must select these items manually.

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

   When 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.

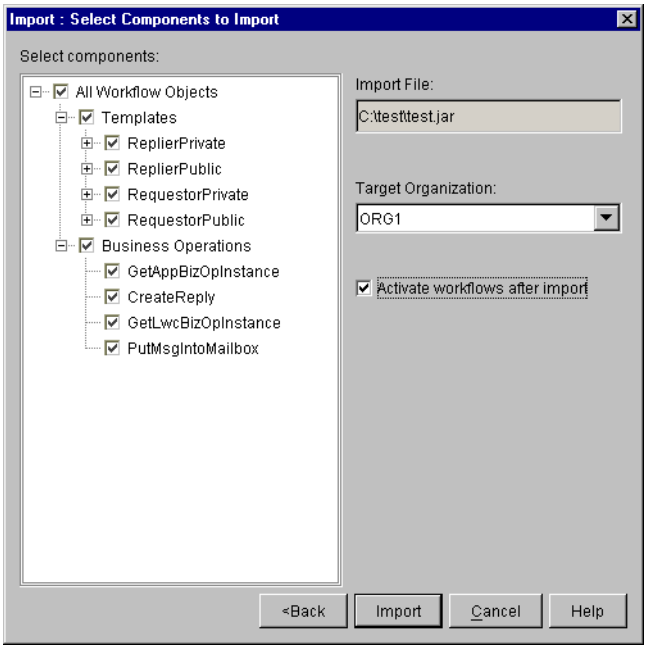
Importing a Workflow Package

To import a workflow package from Studio:

1. Start the Studio client as described in "Starting WebLogic Process Integrator Studio" on page 1-19.
2. Choose Tools→Import Package from the Menu bar.

   The Import: Select File dialog box is displayed.
3. Click Browse, locate the file to import, then click Open.
4. Click Next to display the Import: Select Components to Import dialog box as shown in the following figure.

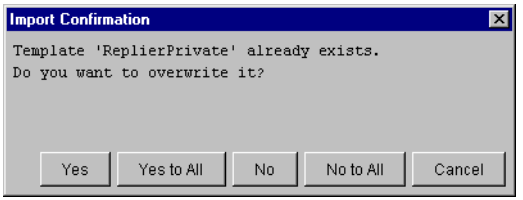
Figure 5-6 Import: Select Components to Import



The dialog lists all items contained in the JAR file.

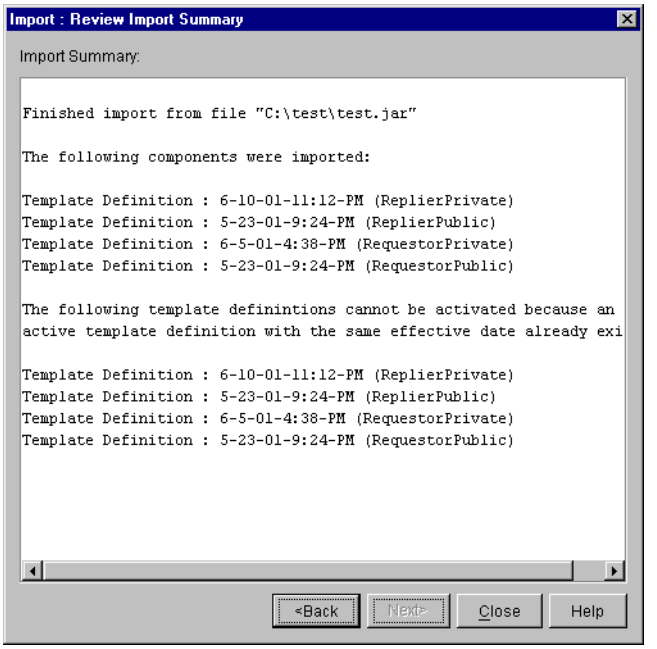
5. Deselect any items you choose not to import.
6. Select the target organization from the drop-down list.
7. To avoid having to activate the workflows after import, check the Activate workflows after import checkbox.
8. Click Import.

   If any items have the same name as a an existing item, the Import Confirmation dialog is displayed as shown in the following figure.



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

Figure 5-7 Import: Review Import Summary



10. Click Close to dismiss the dialog box.

CHAPTER

6 Monitoring WebLogic Collaborate

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

- Overview of Monitoring
- A Note About Conversation Monitoring
- WebLogic Administration Console Monitoring Pages
- Monitoring the Server
- Monitoring Trading Partner Sessions
- Monitoring Delivery Channels
- Monitoring Conversations
- Monitoring Collaboration Agreements
- Monitoring Messages

Overview of Monitoring

The entities that can be controlled and monitored through the WebLogic Collaborate Administration Console are:

- *WebLogic Collaborate*
  The WebLogic Server instance on which WebLogic Collaborate is deployed.
- *Trading partner sessions*
  A trading partner session is a connection between trading partner delivery channels. One or more conversations are associated with this connection.
- *Delivery channels*
  The connection endpoints in a trading partner session.
- *Conversations*
  The exchange of messages associated with the trading partner delivery channels and roles defined in a collaboration agreement.
- *Collaboration agreements*
  The collaboration agreements defined.

A Note About Conversation Monitoring

Although every exchange of messages between trading partners constitutes a conversation, cXML and RosettaNet conversations are not monitored in WebLogic Collaborate.

The XOCP protocol supports a conversation termination protocol that is not supported by cXML or RosettaNet. In XOCP, the initiator of a conversation can terminate the conversation. Other participants have the ability to exit, or leave, the conversation.

The ability to monitor conversations in WebLogic Collaborate is primarily to allow you to take advantage of this feature. The ability to view the number of conversations, view a list of conversations, and individually select a conversation for termination (from the initiating WebLogic Collaborate instance) or exit (from a participant WebLogic Collaborate instance) is limited to the XOCP protocol.

In RosettaNet and cXML, control is 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.

WebLogic Administration Console Monitoring Pages

As described in "WebLogic Collaborate Administration Console Overview" on page 3-2, monitoring functions are available from the Monitoring tab on each of the following WebLogic Collaborate Administration Console pages:

- *WebLogic Collaborate (WLC) page*
  The WLC page is displayed when you select WLC from the navigation tree.
- *Trading partner page*
  The trading partner page is displayed when you select a trading partner from the navigation tree, or select a trading partner from the Trading Partners page.
- *Conversation page*
  The conversation page is displayed when you select a conversation definition from the navigation tree, or select a conversation definition from the Conversations page.
- *Collaboration agreement page*
  The collaboration agreement page is displayed when you select a collaboration agreement from the navigation tree, or select a collaboration agreement from the Collaborations Agreements page.

The following table summarizes the functions available from each page.

Table 6-1 Monitoring WebLogic Collaborate

Select the Monitoring tab on the . . .	Then select this tab . . .	To access these functions . . .
WebLogic Collaborate (WLC) page	General	■ Status (running or inactive). ■ Shutdown options (immediate or terminate).
	Statistics	■ Summary statistics (number of trading partner sessions, active collaborate agreements, active conversations, active delivery channels, number of messages sent and received, time of last message sent and received).
	Log	■ View the WebLogic Collaborate log. ■ Set view options for the log (level of logging, number of lines per page).
Trading partner page	Sessions	■ View a list of the runtime trading partner 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, time of last sent and received message, time of first and last failed message). ■ From the detail of a selected session, link to a list of the conversations or a list of the outstanding messages.
	Delivery Channels	■ View a list of the delivery channels associated with the selected trading partner. ■ View the details of a selected delivery channel (status, trading partner sessions, conversations, collaboration agreements, messages sent). ■ From the detail of a selected delivery channel, modify the status (enable or disable) or link to a list of the trading partner sessions, collaboration agreements, conversations, or messages sent. From each list, additional detail is available.

Table 6-1 Monitoring WebLogic Collaborate (Continued)

Select the Monitoring tab on the . . .	Then select this tab . . .	To access these functions . . .
Conversation page	General	■ View a list of the active conversations for the selected conversation definition (conversation identifier and 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. These features are only available for XOCP. See "A Note About Conversation Monitoring" on page 6-2.
Collaboration agreement page	General	■ View identifying information (name, version, business protocol, associated conversation definition, number of parties) ■ View the status (enabled/disabled, registered/unregistered) ■ Modify the status (enable/disable, register/unregister)

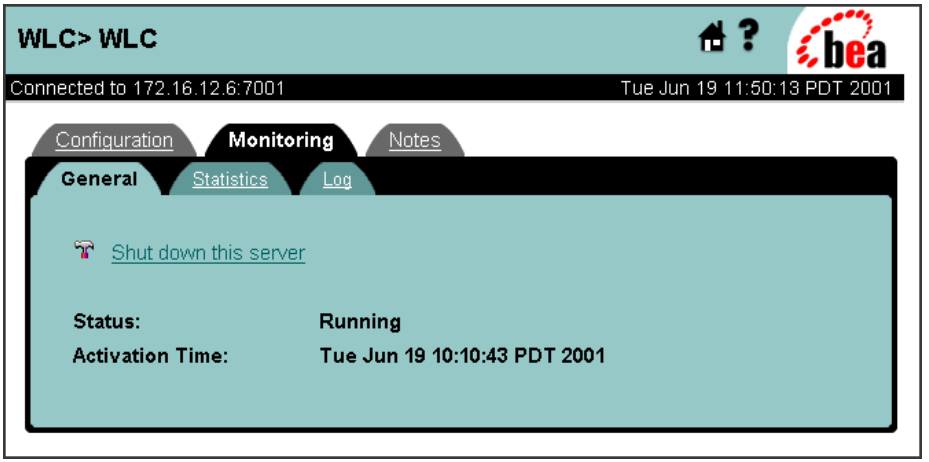
The following sections provide additional information about each type of entity you can monitor from the WebLogic Collaborate Administration Console. The detailed information required to navigate theWebLogic Collaborate Administration Console to view the monitoring options available is provided in the WebLogic Collaborate Administration Console online help.

Note: The information in the online help is also available as a document entitled *BEA WebLogic Collaborate Administration Console Online Help*.

Monitoring the Server

When you select WebLogic Collaborate (WLC) from the navigation tree, and then select the Monitoring tab, the Monitoring General tab is displayed as shown in the following figure:

Figure 6-1 Monitoring the Server



The status and server start time are displayed.

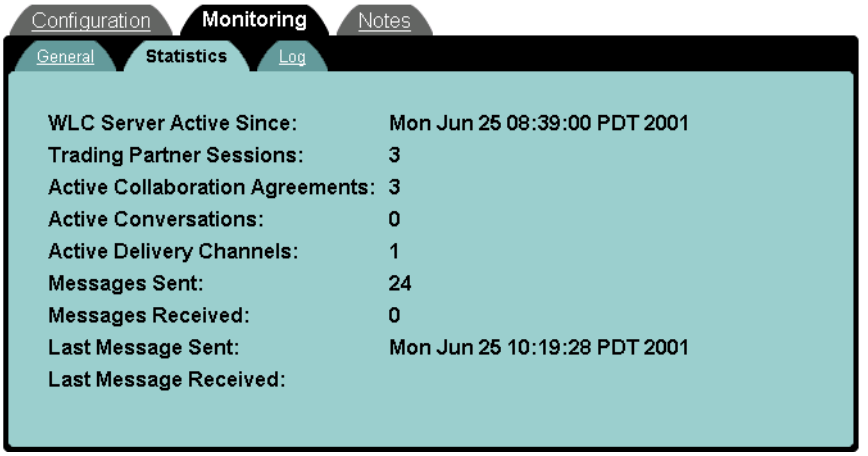
To shut down the server, select `Shut down this server`. The two shut-down options are displayed.

- *Terminate*
  This option shuts down active delivery channels which triggers the termination of the associated trading partner sessions. The termination of the trading partner sessions triggers termination of the associated conversations and removal of queues.
- *Immediate*
  This option stops all activity.

Note: For information about shutting down the WebLogic Collaborate Application, see "Stopping WebLogic Collaborate" on page 1-10.

Select the Statistics tab to display summary statistics as shown in the following figure.

Figure 6-2 Server Statistics



Monitoring Trading Partner Sessions

A trading partner session is a connection between trading partner delivery channels. There are two options for listing active trading partner sessions:

- *List the trading partner sessions for a selected trading partner*
  To list the trading partner sessions for a selected trading partner, select a trading partner from the navigation tree or Trading Partners page, then select the Monitoring tab to view the list of active trading partner sessions for that trading partner.

- *List the trading partner sessions for a selected delivery channel*
  To list the trading partner sessions for a selected delivery channel, select a trading partner from the navigation tree or Trading Partners page, select the Monitoring tab, then select the Delivery Channels tab to display a list of the active delivery channels for that trading partner.

  As described in "Monitoring Delivery Channels" on page 6-9, the number of trading partner sessions for that delivery channel is displayed as part of the summary statistics for the delivery channel. You can view a list of the trading partner sessions for that delivery channel by clicking the number.

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

Figure 6-3 Monitoring a Trading Partner Session

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

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 the trading partner session, select `Shut down this Trading Partner session`. Shut down 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. You can display a list of the active delivery channels for a trading partner as follows:

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 a delivery channel list, the delivery channel status is displayed as shown in the following figure.

Figure 6-4 Monitoring a Delivery Channel

Status and summary statistics for the delivery channel are displayed. From the summary, you can link to a list of the 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 will trigger 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

As discussed in "A Note About Conversation Monitoring" on page 6-2, support for monitoring conversations is only available for XOCP trading partner sessions.

There are three options for listing active XOCP conversations:

- *List the conversations for a selected conversation definition*
  To list the conversations for a selected conversation definition, select a conversation definition from the navigation tree or Conversations page, then select the Monitoring tab to view the list of active conversations for that conversation definition.
- *List the conversations for a selected delivery channel*
  To list the conversations for a selected delivery channel, list delivery channels as described in "Monitoring Delivery Channels" on page 6-9. When you select a delivery channel from the list, the number of conversations for that delivery channel is displayed as part of the summary statistics for the delivery channel. You can view a list of the conversations for that delivery channel by clicking the number.
- *List the conversations for a trading partner session*
  To list the conversations for a selected trading partner session, list trading partner sessions as described in "Monitoring Trading Partner Sessions" on page 6-7. When you select a trading partner session from the list, the number of conversations for that trading partner session is displayed as part of the summary statistics for the trading partner session. You can view a list of the conversations for that trading partner session by clicking the number.

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

Figure 6-5 Monitoring a Conversation

Identifying information, start time, self-initiated indicator, time of last message, and identity of last sender are displayed.

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

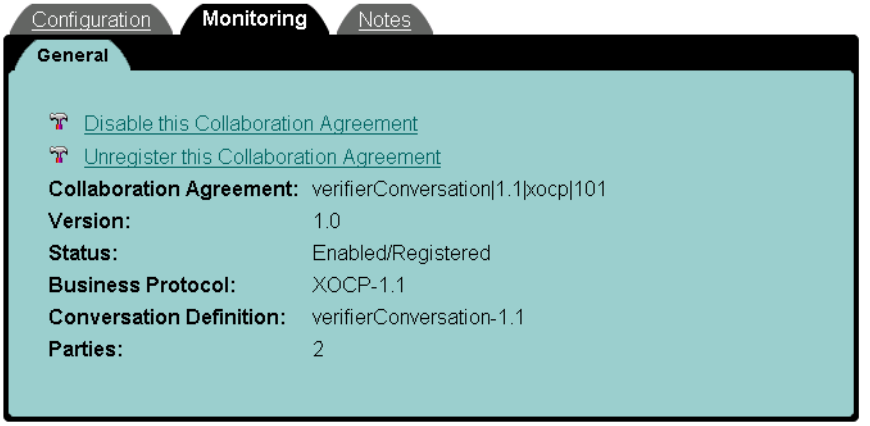
If the local trading partner did not initiate the conversation, the trading partner can leave the conversation by selecting `Leave this Conversation`.

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 the collaboration agreement from the navigation tree or the Collaboration Agreements page, then selecting the monitoring tab.

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

Figure 6-6 Monitoring a Collaboration Agreement



Identifying information, status, business protocol, conversation definition, and number of parties are displayed.

For successful deployment, the collaboration agreement must have two parties, and 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 the delivery channel.).

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

You can disable an enabled collaboration agreement by selecting `Disable this Collaboration Agreement` (displayed for a enabled collaboration agreement) or enable a disabled collaboration agreement by selecting `Enable this Collaboration Agreement` (displayed for a disabled collaboration agreement). Disabling a collaboration agreement will trigger termination of all active conversations associated with the collaboration agreement.

When you disable a collaborationi agreement, existing conversation complete, however, no new conversations for that agreement are started, and no new participants are added to existing converstions.

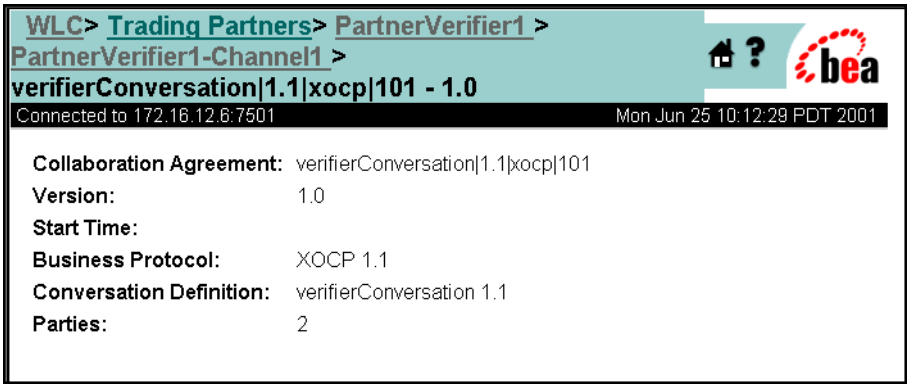
Similarly, you can register an unregistered collaboration agreement or unregister a registered collaboration agreement by selecting `Register this Collaboration Agreement` or `Unregister this Collaboration Agreement` Unregistering a collaboration agreement will trigger termination of all active conversations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

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

1. List delivery channels as described in "Monitoring Delivery Channels" on page 6-9.
2. Select a delivery channel from the list.

   The number of collaboration agreements for that delivery channel is displayed as part of the summary statistics for the delivery channel.
3. View a list of the collaboration agreements for that delivery channel by clicking the number of collaboration agreements.
4. Select a collaboration agreement from the list to display the status as shown in the following figure:

Figure 6-7 Monitoring a Collaboration Agreement



In this case, you are unable to disable or unregister the collaboration agreement.

Monitoring Messages

You can list the outstanding messages for a trading partner session as follows:

1. List trading partner sessions as described in "Monitoring Trading Partner Sessions" on page 6-7.
2. Select a trading partner session from the list.

   When you select a trading partner session from the list, the number of outstanding messages for that trading partner session is displayed as part of the summary statistics for the trading partner session.
3. View a list of the outstanding messages for that trading partner session by clicking the number.

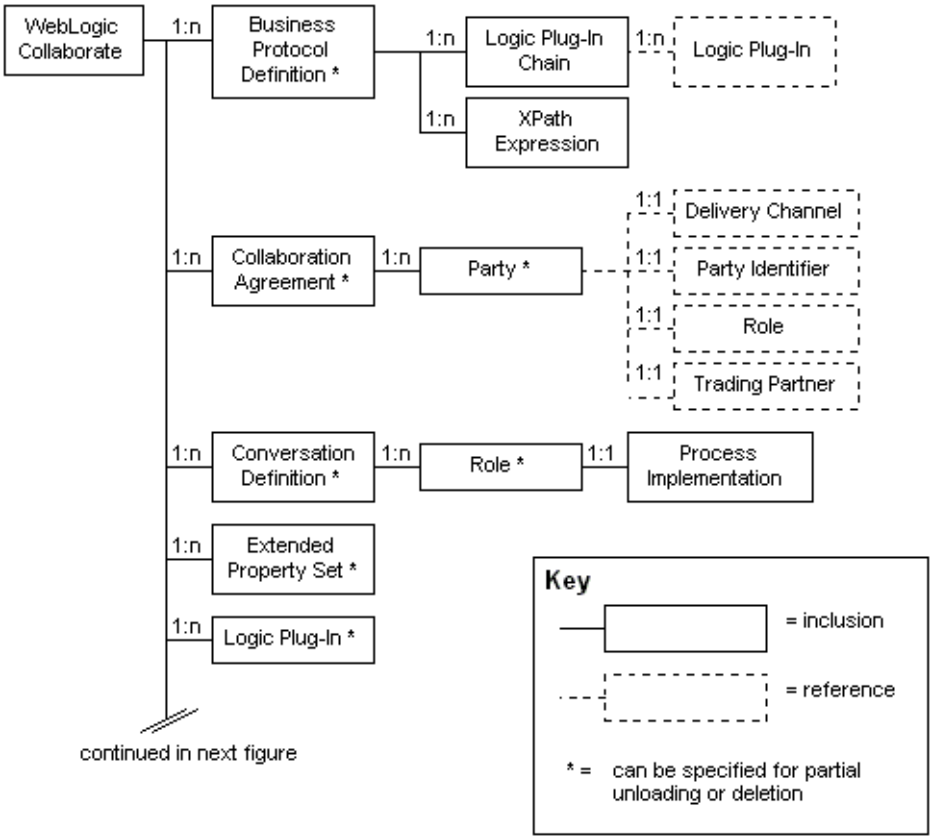
When you select a message from a message list, identifying information, date and time sent, and size are displayed.

CHAPTER

7 Working with the Repository

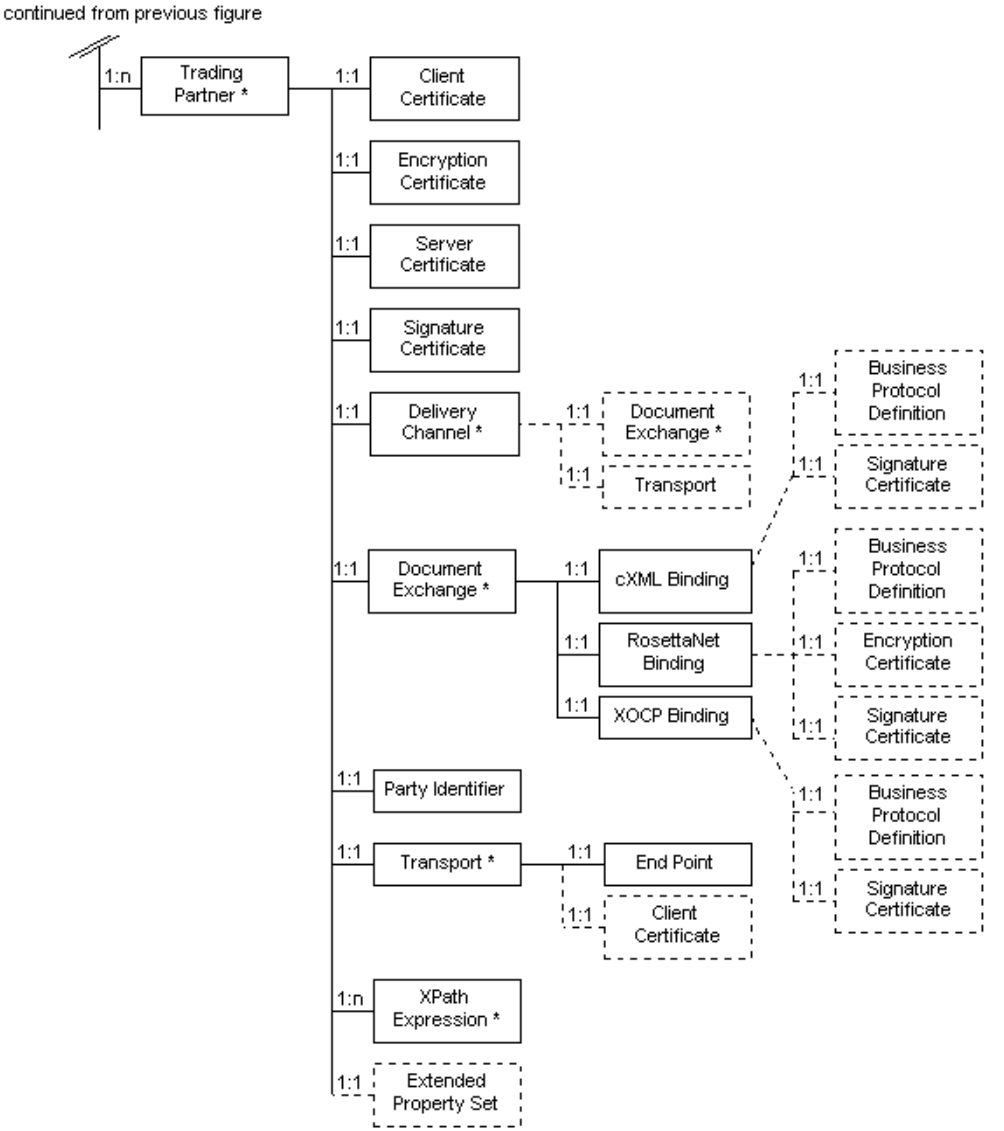
The repository is a database that stores configuration information for WebLogic Collaborate. The following two figures show the relationships among the elements in the repository. For information about configuring the repository data elements, see Chapter 2, “Configuration Requirements.”

Figure 7-1 Elements in the Repository (Part 1)



The following figure is a continuation of the previous figure.

Figure 7-2 Elements in the Repository (Part 2)



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.
- 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 7-1 Elements in the Repository

Element	Description
WebLogic Collaborate	The WebLogic Collaborate element is the root element in the repository. This element represents a WebLogic Collaborate instance. All major elements stem from this root element.
Business protocol definition	A business protocol definition specifies how WebLogic Collaborate processes business messages, including how it reads the messages and how it routes the messages to recipients. A business protocol definition also specifies persistence, retries, and quality of service.
Logic plug-in chain	A logic plug-in chain is a set of logic plug-ins that changes the way in which WebLogic Collaborate routes or filters a message.

Table 7-1 Elements in the Repository (Continued)

Element	Description
XPath expression	An XPath expression is a string written in XPath syntax that, when evaluated, results in one of the following types of objects: ■ Node-set (an unordered collection of nodes without duplicates) ■ Boolean ■ Floating-point number ■ String For information about XPath expressions, see "Expressions" at the following URL: http://www.w3.org/TR/xpath.html#section-Expressions
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.

Table 7-1 Elements in the Repository (Continued)

Element	Description
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	
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	A logic plug-in is value-added software that is installed on WebLogic Collaborate to provide additional processing of the information that passes through WebLogic Collaborate, and that is subject to guidelines and interfaces imposed by WebLogic Collaborate. WebLogic Collaborate provides router logic plug-ins and filter logic plug-ins for each business protocol that it supports. WebLogic Collaborate customers can provide additional functionality by creating custom logic plug-ins that conform to the standards for the business protocol that is being used. Customer-provided logic plug-ins can provide functionality other than routing and filtering, such as billing. For information about logic plug-ins, see *Introducing BEA WebLogic Collaborate* and *Programming BEA WebLogic Collaborate Logic Plug-Ins*.

Table 7-1 Elements in the Repository (Continued)

Element	Description
Trading partner	A trading partner is a business entity that is authorized to send and receive business messages in a conversation.
Certificates	A certificate is a digital certificate. WebLogic Collaborate supports the following types of certificates: ■ Client certificates ■ Encryption certificates ■ Server certificates ■ Signature certificates
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.
Party identifier	A party identifier is a value that specifies a party.
Transport	A transport specifies the properties for a delivery channel's transport level.
End point	An end point is the URL for a trading partner.

8 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

In addition to using the Bulk Loader, you can use the WebLogic Collaborate Administration Console to transfer data to and from the repository, as described in the WebLogic Collaborate Administration Console Online Help. For information about the repository, see Chapter 7, "Working with the Repository."

Understanding the Terminology

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

Table 8-1 Terms Used in This Chapter

This term . . .	Refers to . . .
data element	An element in the repository
XML element	An element in an XML file
attribute	An attribute for an XML element
value	The value of an attribute or the value of a data element

For example, consider the following lines of code from a repository data file:

```
<message-definition name="request.dtd">
    <message-part content-type="text/xml">
        <text-document>request.dtd</text-document>
    </message-part>
</message-definition>
```

In these lines, the following examples of the definitions are used:

- `message-definition` is an XML element that specifies a `message-definition` data element.
- `content-type` is an attribute.
- `text/xml` is the value for the `content-type` attribute.
- `request.dtd` is the value for the `text-document` data element.

Importing Data into the Repository

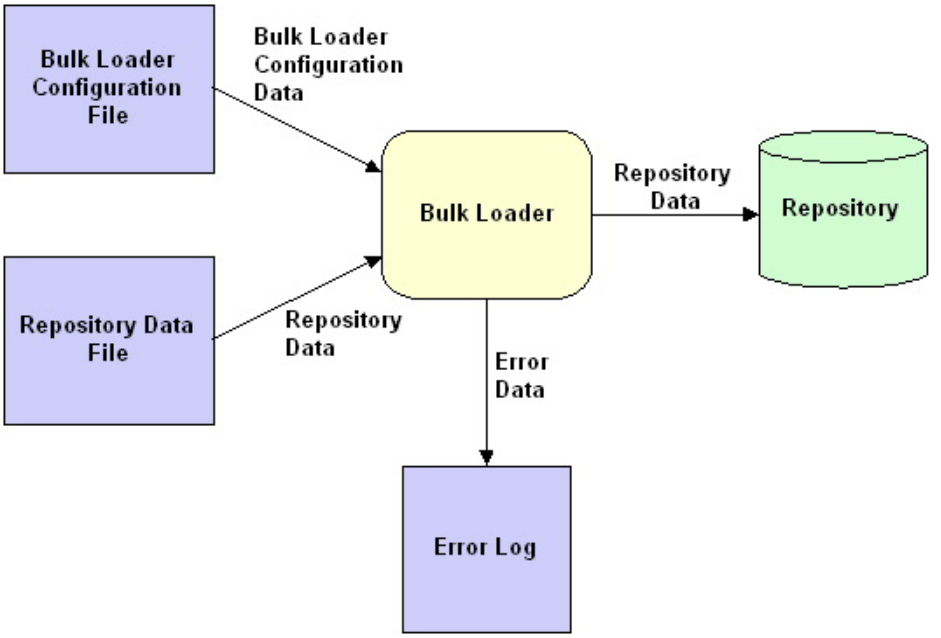
The following sections provide information about importing repository data:

- 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 8-1 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 WebLogic Collaborate root element specified in the repository data file does not exist in the repository, the Bulk Loader creates the WebLogic Collaborate root element.
2. If the WebLogic Collaborate 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:
   a. If the data element exists in the repository and has the same data element values, the same attributes, and the same attribute values as the corresponding XML element in the repository data file, the Bulk Loader does not do anything.
   b. 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 each attribute.
   c. 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 each attribute.

Table 8-2 Logic for Processing an Attribute

Does the Value 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 8-17.

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

Table 8-3 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

Procedure for Importing Data into the Repository

Note: In addition to using the Bulk Loader, you can use the WebLogic Collaborate Administration Console to transfer data to and from the repository, as described in the WebLogic Collaborate Administration Console Online Help.

Note: You cannot run the Bulk Loader when WebLogic Collaborate is running.

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 8-13.

2. Create a repository data file.

   For information about creating a repository data file, see "Working with the Repository Data File" on page 8-15.

3. To import data from the repository data file into the repository, enter one of the following commands:

   - UNIX: `bulkloader.sh` *`cfg_file`*
   - Windows: `bulkloader` *`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 8-17.

Exporting Data from the Repository

The following sections provide information about exporting repository data:

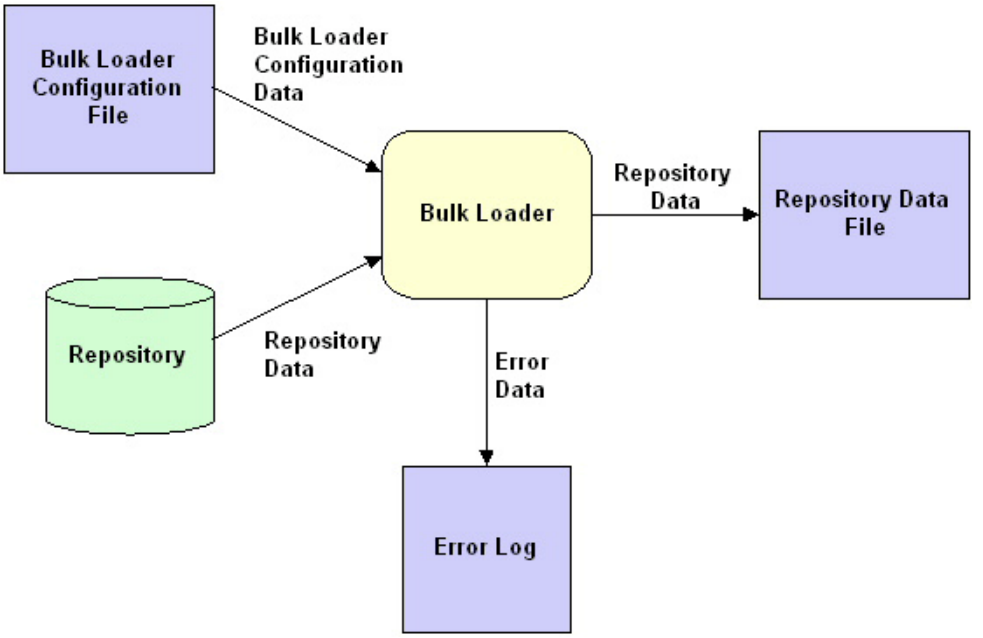
- How the Bulk Loader Exports Data
- Full and Partial Repository Exports
- Short and Long Repository Exports
- Procedure for Exporting Repository Data

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 8-2 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 8-4 Full and Partial Repository Exports

Type of Export	Description
Full	The Bulk Loader exports all the data.
Partial	The Bulk Loader exports a subset of the data.

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 `WLCConfig.dtd` in the `dtd` subdirectory of your WebLogic Collaborate installation directory.

You can identify a specific data element instance in the Bulk Loader configuration file. For example, to export a specific message definition, specify the name of the WebLogic Collaborate element and the name of the message definition, as shown in the following listing. This strategy applies to all types of data elements.

Listing 8-1 Example of a Bulk Loader Configuration File for Exporting a Specific Message Definition

```
<?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 name="office_supplies">
                <message-definition name="request.dtd">
                </message-definition>
            </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 8-5 Short and Long Repository Exports

Type of Export	Description
Short (standard)	The Bulk Loader organizes the output data to represent the user's view of the repository.
Long (extensive)	The Bulk Loader organizes the output data 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 8-3. For a complete description of the `format` attribute, see `WLCConfig.dtd` in the `dtd` subdirectory of your WebLogic Collaborate installation directory.

Procedure for Exporting Repository Data

Note: In addition to using the Bulk Loader, you can use the WebLogic Collaborate Administration Console to transfer data to and from the repository, as described in the WebLogic Collaborate Administration Console Online Help.

Note: You cannot run the Bulk Loader when WebLogic Collaborate is running.

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 8-13.

2. To export data from the repository to a repository data file, enter one of the following commands:

   - UNIX: `bulkloader.sh` *`cfg_file`*
   - Windows: `bulkloader` *`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 8-17.

Deleting Data from the Repository

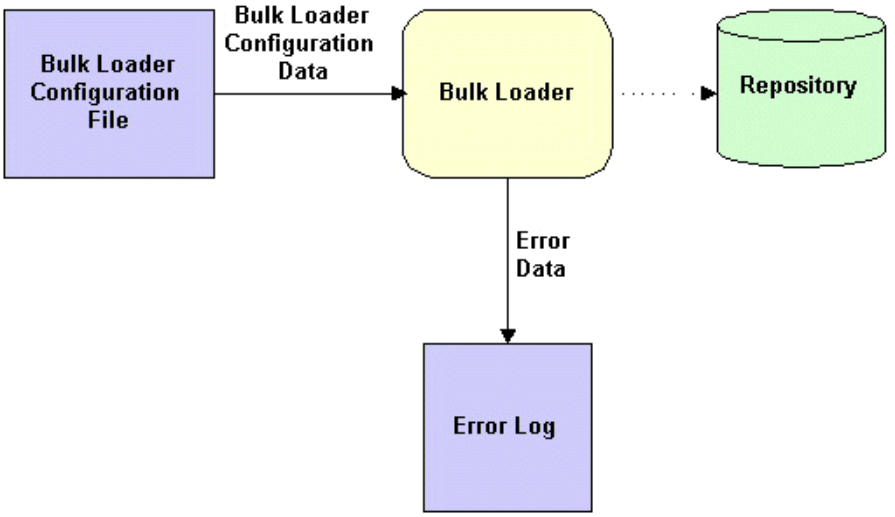
The following sections provide information about deleting repository data:

- 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 8-3 Using the Bulk Loader to Delete Repository Data



Procedure for Deleting Repository Data

Note: In addition to using the Bulk Loader, you can use the WebLogic Collaborate Administration Console to transfer data to and from the repository, as described in the WebLogic Collaborate Administration Console Online Help.

Note: You cannot run the Bulk Loader when WebLogic Collaborate is running.

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 8-6.
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 8-13.
3. To delete data from the repository, enter one of the following commands:
   - UNIX: `bulkloader.sh` *`cfg_file`*
   - Windows: `bulkloader` *`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 8-17.

Working with the Bulk Loader Configuration File

The Bulk Loader configuration file contains the database login information and processing instructions for the Bulk Loader. The Bulk Loader configuration file is an XML file that uses the `WLCConfig.dtd` file, which is in the `dtd` subdirectory of your WebLogic Collaborate installation directory.

To create a Bulk Loader configuration file:

1. Create an XML file that specifies `WLCConfig.dtd` as the DTD file.
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 each type of Bulk Loader task:

- 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 8-2 Example of a Bulk Loader Configuration File for Importing Data into the Repository

```
<?xml version="1.0"?>
<!DOCTYPE wlc-config SYSTEM "WLCConfig.dtd">
<wlc-config>
    <load-processing-parameters database-initialization="no"\
```

```
        transaction-level="all">
        <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>ImportRepoData.xml</xml-file-name>
    </load-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:

- `load-processing-parameters`—This XML element tells the Bulk Loader to import data from the repository data file into the repository.
- `database-initialization`—The value for this attribute specifies whether the Bulk Loader deletes all data from the repository before performing the repository import. The possible 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. The possible values are `all` and `default`. For more information, see "Checking Data Integrity" on page 8-18.
- `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.

Bulk Loader Configuration File for Exporting Data

The following listing is an example Bulk Loader configuration file for exporting data from the repository.

Listing 8-3 Example of a Bulk Loader Configuration File for Performing a Full Export from the Repository

```
<?xml version="1.0"?>
<!DOCTYPE wlc-config SYSTEM "WLCConfig.dtd">
```

```
<wlc-config>
    <unload-processing-parameters format="long">
        <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>
    </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:

- `unload-processing-parameters`—This XML element tells the Bulk Loader to export data from the repository to the repository data file.
- `format`—The value for this attribute specifies whether the Bulk Loader exports data in the long (extensive) format or in the short (standard) format. The possible values are `short` and `long`. For more information, see "Short and Long Repository Exports" on page 8-9.
- `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.

Working with the Repository Data File

The repository data file is an XML file that uses the `WLC.dtd` file, which is in the `dtd` subdirectory of your WebLogic Collaborate installation directory. To create a repository data file, create an XML file that specifies `WLC.dtd` as the DTD file.

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

Listing 8-4 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 8-6 Timestamp Appended to Name of Existing Log File

This string . . .	Indicates the . . .
yyyy	Year
mm	Month in numeric format (between 01 - 12)
dd	Day in numeric format
hh	Hour (between 00 - 23)
mi	Minute
ss	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 8-7 Files Validated by the Bulk Loader

Task	Validated Files
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 8-2. 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 WebLogic Collaborate data element at the WebLogic Collaborate level (such as a WebLogic Collaborate attribute), it rolls back all transactions that have been performed for that WebLogic Collaborate 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.

CHAPTER

9 Configuring Persistence and Recovery

The following sections describe persistence and recovery:

- Understanding Persistent Mode
- Understanding Nonpersistent Mode
- Understanding Recovery
- Configuring Persistence and Recovery

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, WebLogic Collaborate stores the states of the following types of run-time objects in persistent storage:

- System
- Delivery channels
- Trading partners
- Conversations

- Messages
- Message queues
- Tables related to the preceding objects

Persistent storage consists of a database that WebLogic Collaborate accesses by means of a connection pool. The same database is used by both persistent storage and the repository. WebLogic Collaborate 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.

Multiple objects can change state during the time that a message passes through WebLogic Collaborate. WebLogic Collaborate updates the persistent storage for these objects as a group. If a message passes through WebLogic Collaborate successfully, then the recorded changes are retained. If the message processing fails in any part of WebLogic Collaborate, then WebLogic Collaborate can retry some operations. When an operation fails, WebLogic Collaborate discards the changes that the operation caused. To maintain and update a set of objects states as a group, WebLogic Collaborate uses transactions.

WebLogic Collaborate performs special processing for messages. WebLogic Collaborate uses the message size and the large message support configuration to 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, WebLogic Collaborate 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, WebLogic Collaborate saves the entire message in persistent storage.

For information about the large message threshold and large message support, see "Configuring WebLogic Collaborate" in *BEA WebLogic Collaborate Administration Console Online Help*.

Understanding Nonpersistent Mode

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

Note: It is recommended that you use this option only when you are developing your WebLogic Collaborate system, not during run time.

Note: If you migrated a Java messaging application that was written using the WebLogic Collaborate C-Enabler API to WebLogic Collaborate Release 2.0, the migrated application must be run in a separate JVM in nonpersistent mode.

Understanding Recovery

The following list describes the WebLogic Collaborate phases that affect recovery:

1. Startup

   WebLogic Collaborate starts in either persistent mode or nonpersistent mode. You cannot change the persistence mode while WebLogic Collaborate is running. For information about configuring the persistence mode, see "Configuring Persistence and Recovery" on page 9-5.

2. Shutdown

   WebLogic Collaborate shuts down in one of the following modes:

   - *Terminate*—WebLogic Collaborate shuts down delivery channels and trading partner sessions, terminates conversations, and exits.
   - *Immediate*—WebLogic Collaborate exits immediately. This behavior is similar to a crash.

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

3. Restart

   When WebLogic Collaborate restarts after a shutdown, its behavior depends on whether it is running in persistent mode or nonpersistent mode:

   - If WebLogic Collaborate starts in nonpersistent mode, it does not undertake any particular recovery activities.
   - If WebLogic Collaborate is in persistent mode, it tries to read state records from persistent storage to determine whether or not to perform the recovery procedure. If WebLogic Collaborate shuts down in terminate mode, no state records are found. Otherwise, WebLogic Collaborate 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 WebLogic Collaborate was down.

     c. Restarts message queues for active trading partners.

Configuring Persistence and Recovery

The following table describes how to set the persistence mode when you start WebLogic Collaborate.

Table 9-1 Setting the Persistence Mode

When you start WebLogic Collaborate with the following method . . .	Do the following to start WebLogic Collaborate in persistent mode . . .	Or do the following to start WebLogic Collaborate in nonpersistent mode . . .
WebLogic Collaborate Administration Console For information about starting WebLogic Collaborate from the WebLogic Collaborate Administration Console, see Chapter 1, "Starting, Stopping, and Customizing WebLogic Collaborate."	1. Go to the Monitoring → General tab in the WebLogic Collaborate Administration Console. 2. Click `Start this server`. 3. The resulting dialog box presents a checkbox labeled Persistence On. Select the checkbox if it is not already selected. 4. Click Yes.	1. Go to the Monitoring → General tab in the WebLogic Collaborate Administration Console. 2. Click `Start this server`. 3. The resulting dialog box presents a checkbox labeled Persistence On. De-select the checkbox if it is not already de-selected. 4. Click Yes. **Note:** It is recommended that you run in nonpersistent mode only when you are developing your WebLogic Collaborate system, not during run time.

Table 9-1 Setting the Persistence Mode (Continued)

When you start WebLogic Collaborate with the following method . . .	Do the following to start WebLogic Collaborate in persistent mode . . .	Or do the following to start WebLogic Collaborate in nonpersistent mode . . .

Start menu	In the Start menu, select the Start Server command for WebLogic Collaborate. For details about how to do this, see Chapter 1, "Starting, Stopping, and Customizing WebLogic Collaborate."	1. Add the `PERSISTENCE` value to the `config.xml` file. `PERSISTENCE` is one of the possible values for the `StartupClass`'s `Arguments` attribute. For example:
		`<StartupClass`
		`ClassName=`
		`"com.bea.b2b.`
		`server.Startup"`
		`Name=`
		`"WLCStartup"`
		`Arguments=`
		`"PERSISTENCE=OFF"`
		`Targets=`
		`"myserver"`
		`/>`
		2. In the Start menu, choose the Start Server command for WebLogic Collaborate. For details about how to do this, see Chapter 1, "Starting, Stopping, and Customizing WebLogic Collaborate."

Table 9-1 Setting the Persistence Mode (Continued)

When you start WebLogic Collaborate with the following method . . .	Do the following to start WebLogic Collaborate in persistent mode . . .	Or do the following to start WebLogic Collaborate in nonpersistent mode . . .
Command line	Issue the command to start WebLogic Collaborate. For instructions, see Chapter 1, "Starting, Stopping, and Customizing WebLogic Collaborate."	1. Add the `PERSISTENCE` value to the `config.xml` file. `PERSISTENCE` is one of the possible values for the `StartupClass`'s `Arguments` attribute. For example: `<StartupClass ClassName= "com.bea.b2b. server.Startup" Name= "WLCStartup" Arguments= "PERSISTENCE=OFF" Targets= "myserver" />` 2. Issue the command to start WebLogic Collaborate. For instructions, see Chapter 1, "Starting, Stopping, and Customizing WebLogic Collaborate."

A WebLogic Collaborate Sample Configuration Files

This topic provides system administrators with sample configuration files for BEA WebLogic Collaborate. During installation, these files are automatically customized, so they will work on any supported computer system and network. They are provided here, along with descriptions of the information required in each.

The sample configuration files are provided in teh following sections:

- config.xml
- setEnv.cmd/setEnv.sh
- startWebLogic.cmd/startWeblogic.sh
- fileRealm.properties

Each file includes sample values for parameters that are updated by the installation program. These sample values are highlighted in **bold**.

config.xml

By default, the BEA WebLogic Collaborate installation sets up the `collaborate/config/samples` and `collaborate/config/mydomain` domains. The configuration for each domain is defined in eXtensible Markup Language (XML). Persistent storage for a domain configuration is provided by the `config.xml` file located in *`install_dir`*`/collaborate/config/`*`domain_name`*, where *`install_dir`* is the directory under which WebLogic Integration has been installed (typically <*`BEA_Home`*>`/wlintegration2.0`).

This section presents and describes the contents of the `collaborate/config/mydomain/config.xml` file

This sample is provided for informational purposes only. The `config.xml` file should not be edited directly. Any changes made to the file while the domain is active will not have any effect on the domain configuration and will be overwritten by the boot version of the file.

You can view and modify the settings that are captured in the `config.xml` through the WebLogic Server Administration Console. When you start the WebLogic Server Administration Console as described in "Starting the WebLogic Server Administration Console" on page 1-17, the server home page is displayed. Items in the navigation tree in the left-pane correspond to the elements and attributes defined in the `config.xml` file.

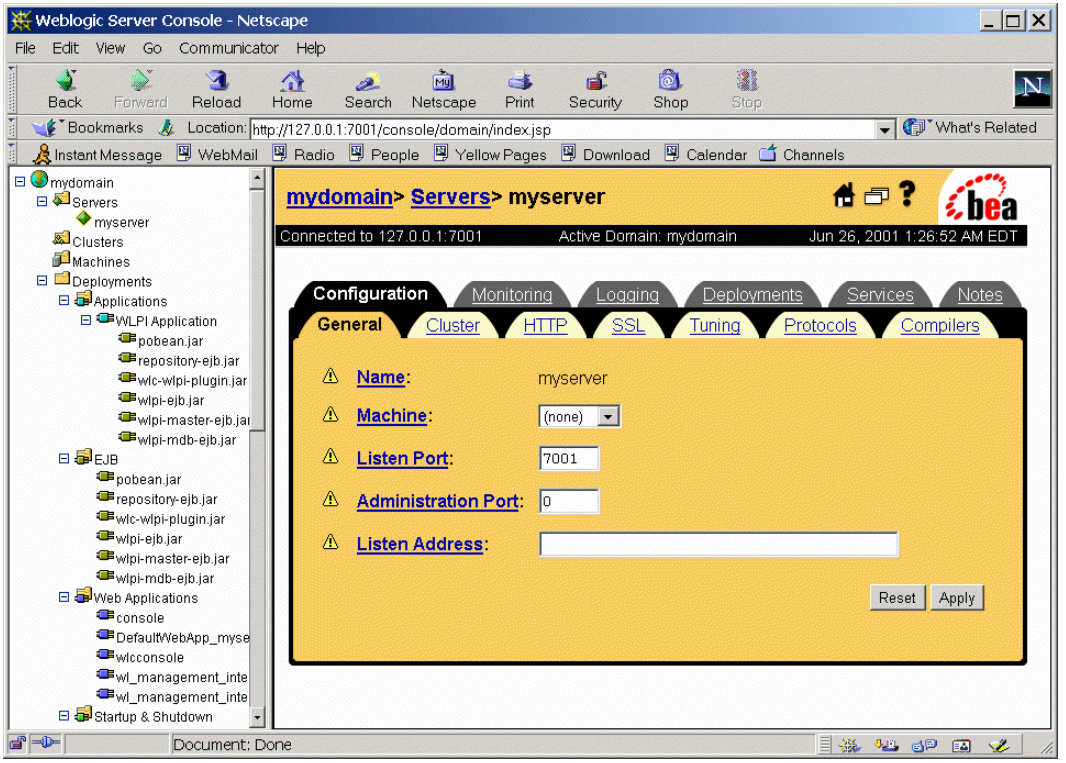
The root element of the `config.xml` file is Domain. The first entry in the domain is the server configuration, as shown in the following figure.

Figure A-1 The config.xml File: Part 1

```
<Domain Name="mydomain">

<Server
   ListenPort="7001"
   Name="myserver"
   ThreadPoolSize="15"
   TransactionLogFilePrefix="./config/mydomain/"
   NativeIOEnabled="true">
      <Log FileName="myserver.log" Name="myserver"/>
      <WebServer DefaultWebApp="DefaultWebApp_myserver"
         HttpsKeepAliveSecs="120" KeepAliveSecs="60"
         LogFileName="./config/mydomain/access.log"
         LoggingEnabled="true" Name="myserver"/>
      <Log FileMinSize="1024" Name="config/mydomain/myserver"/>
      <KernelDebug Name="myserver"/>
      <ServerDebug Name="myserver"/>
      <SSL Name="myserver"/>

   <!--  Include following in Server to see log info on screen
   StdoutDebugEnabled="true"
   StdoutEnabled="true"
   StdoutSeverityLevel="64"
   -->
</Server>
```

The Domain and Server elements and attributes define the basic configuration. The settings here are reflected in the WebLogic Server Administration Console. For example, if you select myserver from the navigation tree, the server page is displayed as shown in the following figure.

Figure A-2 WebLogic Server Administration Console Server Page



By clicking the appropriate tab on this page, you can update the listen port, log file name, log level and output, and other server configuration parameters. For help with any settings, click the question mark in the upper right to view the online help, or see the WebLogic Server documentation set at the following URL:

http://e-docs.bea.com/wls/docs60

As shown in the following figure, the server element is followed by the specification of the log file, and settings for the JMS JDBC Store, Java Transaction API (JTA), Java Messaging Server (JMS) server, and security.

Figure A-3 The config.xml File: Part 2

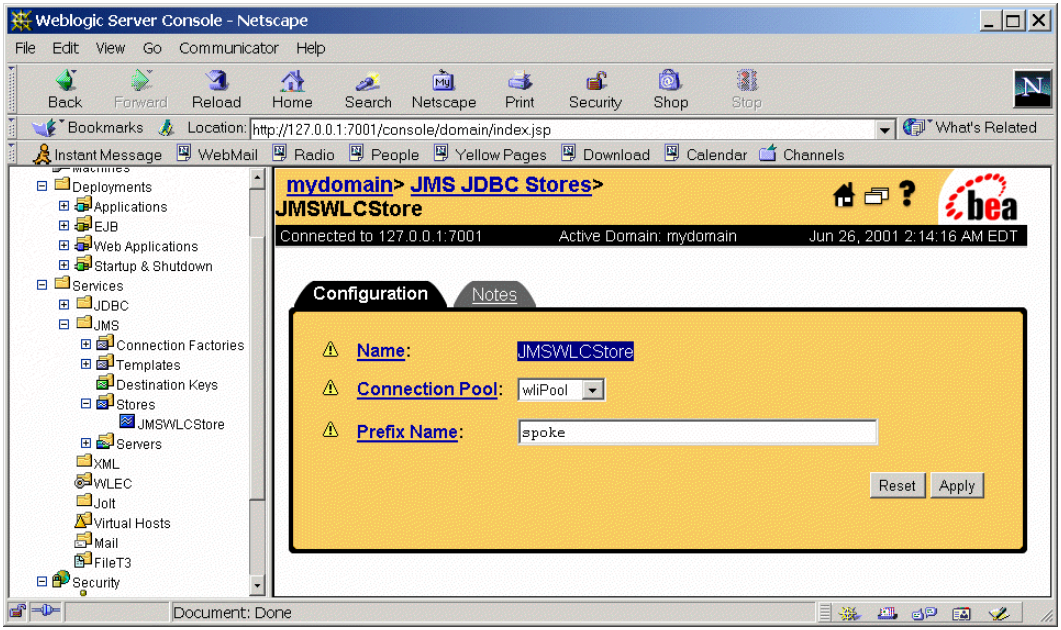
```
<Log
   FileName="config/mydomain/mydomain.log"
   Name="myserver"/>

<JMSJDBCStore
   ConnectionPool="wliPool"
   Name="JMSWLCStore"
   PrefixName="spoke"/>

<JMSServer
   Name="WLCJMSServer"
   Targets="myserver"
   Store="JMSWLCStore" />

<JTA
   Name="mydomain"
   TimeoutSeconds="3600"/>

<ApplicationManager
   Name="mydomain"/>

<Security
   GuestDisabled="false"
   Name="mydomain"
   PasswordPolicy="mypasswordpolicy"
   Realm="myRealm"/>

<PasswordPolicy
   MinimumPasswordLength="4"
   Name="mypasswordpolicy"/>

<Realm
   CachingRealm=""
   FileRealm="myFileRealm" Name="myRealm"/>

<FileRealm
   Name="myFileRealm"/>
```

For optimum performance, a fully qualified PrefixName should be specifed as follows: [[*catalog.*]*schema.*]*prefix*

To optimize performance, the setting for the JMS JDBC store prefix name should be customized for your database. The syntax for prefix name is [[*catalog.*]*schema.*]*prefix*].

To update the JMS store prefix name, select Services→JMS→Stores→JMSWLCStore. The JMSWLCStore page is displayed as shown in the following figure.

Figure A-4 JMS JDBC Store Page



As shown in the following figure, the next section of the `config.xml` file deploys the web applications required for the WebLogic Collaborate Administration Console, and the WebLogic Collaborate startup and shut down classes.

Figure A-5 The config.xml File: Part 3

```
<Application
   Deployed="true"
   Name="DefaultWebApp_myserver"
   Path="./config/mydomain/applications">
      <WebAppComponent
         Name="DefaultWebApp_myserver"
         Targets="myserver"
         URI="DefaultWebApp_myserver"
         WebServers="myserver"
         IndexDirectoryEnabled="true"/>
</Application>

<Application
   Name="WLCConsole"
   Path="./config/mydomain/applications">
      <WebAppComponent
         Name="wlcconsole"
         ServletReloadCheckSecs="1"
         Targets="myserver"
         URI="wlcconsole.war"/>
</Application>

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

<ShutdownClass
   ClassName="com.bea.b2b.server.Shutdown"
   Arguments="mode=TERMINATE"
   Name="WLCShutdown"
   Targets="myserver"/>
```

As shown in the following figure, the next section of the `config.xml` file includes JMS connection information for WebLogic Process Integration, and the JDBC connection pool information. The bold text indicates settings that are updated based on the information you provide during installation. The arrow indicates the continuation of text that appears on a single line in the file.

For information about updating the JDBC configuration, see "Updating the JDBC Connection Pool" on page 1-27.

Figure A-6 The config.xml File: Part 4

```
<JMSJDBCStore
   ConnectionPool="wliPool"
   Name="JMSWLCStore"/>

<JMSServer
   Name="WLCJMSServer"
   Targets="myserver"
   Store="JMSWLCStore"/>

<JDBCTxDataSource
   JNDIName="WLCHub.DS"
   Name="WLCHub.DS"
   PoolName="wliPool"
   EnableTwoPhaseCommit="true"
   Targets="myserver"/>

<JDBCConnectionPool
   CapacityIncrement="1"
   DriverName="weblogic.jdbc.mssqlserver4.Driver"
   InitialCapacity="10"
   LoginDelaySeconds="1"
   MaxCapacity="100"
   Name="wliPool"
   Properties="user=myusername;password=mypassword"
   RefreshMinutes="0"
   ShrinkPeriodMinutes="15"
   ShrinkingEnabled="true"
   Targets="myserver"
   URL="jdbc:weblogic:mssqlserver4:WLIDB@rdbmshost:1433"/>

<StartupClass
   ClassName="com.bea.wlpi.server.wlpiinit.WLPIInit"
   Name="WLPIInit"
   Arguments="poolName=wliPool,user=wlpisystem,pwd=wlpisystem,
    expireInterval=30000,deliverInterval=86400000,checkInterval=600000"
   Targets="myserver"/>

<JMSConnectionFactory
   AllowCloseInOnMessage="true"
   JNDIName="com.bea.wlpi.QueueConnectionFactory"
   Name="wlpiQueueFactory"
   Targets="myserver"
   UserTransactionsEnabled="true"/>

<JMSConnectionFactory
   AllowCloseInOnMessage="true"
   JNDIName="com.bea.wlpi.TopicConnectionFactory"
   Name="wlpiFactory"
   Targets="myserver"
   UserTransactionsEnabled="true"/>
```

Bold indicates values updated based on information provided during installation.

The next section deploys the EJBs required for WebLogic Process Integrator. The deployment order is controlled by the `DeploymentOrder` setting. You can use this setting as required to control deployment order for custom applications and EJBs.

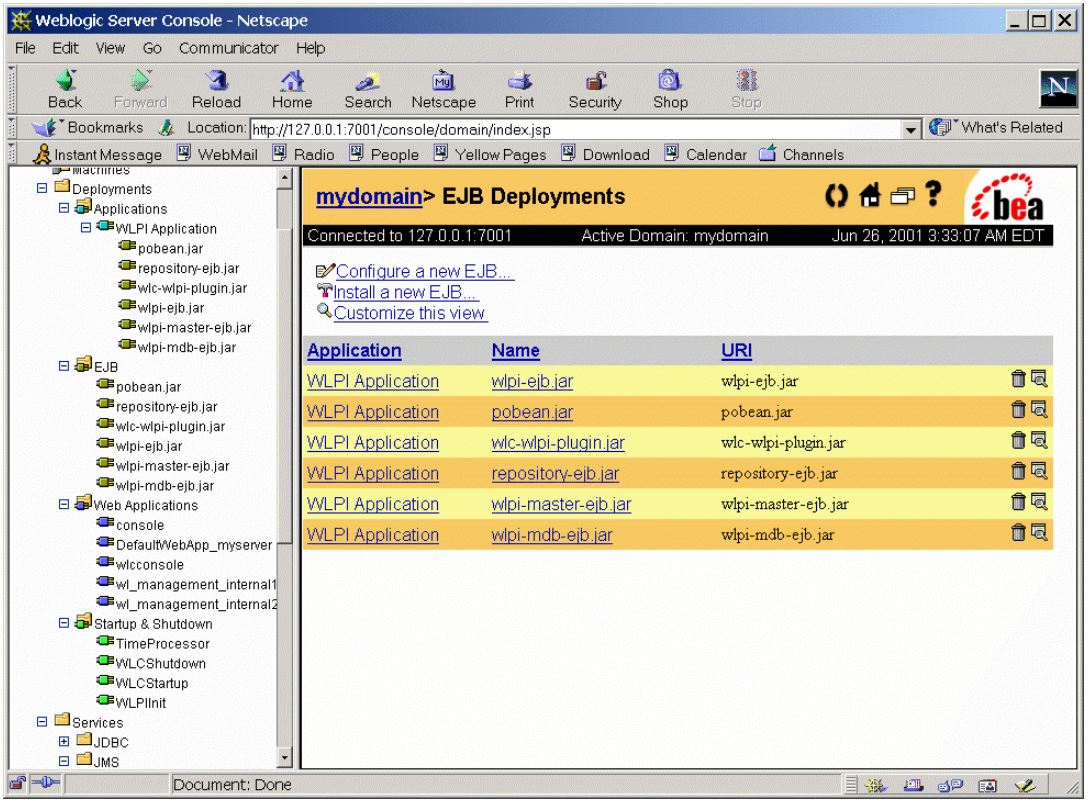
Figure A-7 The config.xml File: Part 5

```
<Application Name="WLPI Application"
   Path="c:\bea\wlintegration2.0\processintegrator\lib">
      <EJBComponent
         Name="wlpi-ejb.jar"
         Targets="myserver"
         URI="wlpi-ejb.jar"
         DeploymentOrder="1"/>
      <EJBComponent
         Name="wlpi-master-ejb.jar"
         Targets="myserver"
         URI="wlpi-master-ejb.jar"
         DeploymentOrder="2"/>
      <EJBComponent
         Name="wlpi-mdb-ejb.jar"
         Targets="myserver"
         URI="wlpi-mdb-ejb.jar"
         DeploymentOrder="3"/>
      <EJBComponent
         Name="repository-ejb.jar"
         Targets="myserver"
         URI="repository-ejb.jar"
         DeploymentOrder="0"/>
      <EJBComponent
         Name="wlc-wlpi-plugin.jar"
         Targets="myserver"
         URI="wlc-wlpi-plugin.jar"
         DeploymentOrder="4"/>
      <EJBComponent
         Name="pobean.jar"
         Targets="myserver"
         URI="pobean.jar"
         DeploymentOrder="5"/>
</Application>
```

Controls deployment order

You can view and update parameters for the applications, Web applications, and EJBs deployed by selecting Deployments from the WebLogic Server Administration Console navigation tree, then selecting the appropriate deployment type. The EJB deployments are shown in the following figure.

Figure A-8 Deployments



As shown in the following figure, the next section of the `config.xml` file sets up the JMS Topics required by WebLogic Process Integrator.

Figure A-9 The config.xml File: Part 6

```
<JMSServer
   Name="WLCJMSServer"
   Targets="myserver"
   TemporaryTemplate="TemporaryTemplate">
      <JMSTemplate Name="TemporaryTemplate"/>
      <JMSTopic
         JNDIName="com.bea.wlpi.TimerTopic"
         Name="wlpiTimer"/>
      <JMSQueue
         JNDIName="com.bea.wlpi.EventQueue"
         Name="eventQueue"/>
      <JMSQueue
         JNDIName="com.bea.wlpi.ValidatingEventQueue"
         Name="wlpivalidatingEvent"/>
      <JMSTopic
         JNDIName="com.bea.wlpi.ErrorTopic"
         Name="wlpiError"/>
      <JMSTopic
         JNDIName="com.bea.wlpi.Timer"
         Name="Timer" StoreEnabled="default"/>
      <JMSTopic
         JNDIName="com.bea.wlpi.AuditTopic"
         Name="wlpiAudit"/>
      <JMSQueue
         JNDIName="com.bea.wlpi.InitQueue"
         Name="initQueue"/>
      <JMSTopic
         JNDIName="com.bea.wlpi.
         EventTopic" Name="wlpiEvent"/>
      <JMSTopic
         JNDIName="com.bea.wlpi.NotifyTopic"
         Name="wlpiNotify"/>
</JMSServer>

<JDBCTxDataSource
   EnableTwoPhaseCommit="true"
   JNDIName="com.bea.wlpi.TXDataSource"
   Name="TXDataSource"
   PoolName="wliPool"
   Targets="myserver"/>
```

As shown in the following figure, the next section includes the startup class for the WebLogic Process Integrator time processor, and settings for the WebLogic Process Integrator RDBMS Realm security configuration. The values in bold are updated during installation based on the information you provide. The arrows indicate the continuation of text that appears on a single line in the file.

Figure A-10 The config.xml File: Part 7

```
<StartupClass
   ClassName="com.bea.wlpi.server.timeprocessor.TimeProcessor"
   Name="TimeProcessor"
   Arguments="poolName=wliPool,user=wlpisystem,pwd=wlpisystem,
    connectionFactory=com.bea.wlpi.TopicConnectionFactory,
    topic=com.bea.wlpi.TimerTopic"
   Targets="myserver"/>


<RDBMSRealm
   RealmClassName="com.bea.wlpi.rdbmsrealm.RDBMSRealm"
   Name="wlpiRDBMSRealm"
   DatabaseDriver="weblogic.jdbc.mssqlserver4.Driver"
   DatabasePassword="mypassword"
   DatabaseURL="jdbc:weblogic:mssqlserver4:WLIDB@rdbmshost:1433"
   DatabaseUserName="myusername"
   SchemaProperties="getGroupNewStatement=false;removeUserFromGroup=DELETE
FROM USERMEMBER WHERE USERID = ? AND GROUPID = ?;getAcls=SELECT NAME,
PRINCIPAL, PERMISSION FROM aclentries ORDER BY NAME,
PRINCIPAL;addUserToGroup=INSERT INTO USERMEMBER (USERID, GROUPID) VALUES (
?, ? );getGroupMembersUsers=SELECT USERMEMBER.USERID, PASSWORD FROM
USERMEMBER, WLSUSER WHERE GROUPID = ? AND USERMEMBER.USERID =
WLSUSER.USERID;newGroup=INSERT INTO WLSGROUP (GROUPID) VALUES ( ?
);addGroupToGroup=INSERT INTO GROUPMEMBER (GROUPMEMBERID, GROUPID) VALUES
( ?, ? );newUser=INSERT INTO WLSUSER (USERID, PASSWORD) VALUES ( ? , ?
);removeGroupFromGroup=DELETE FROM GROUPMEMBER WHERE GROUPMEMBERID = ? AND
GROUPID = ?;deleteGroup4=DELETE FROM WLSGROUP WHERE GROUPID =
?;deleteUser3=DELETE FROM WLSUSER WHERE USERID = ?;deleteGroup3=DELETE
FROM USERMEMBER WHERE GROUPID = ?;getPermissions=SELECT DISTINCT
PERMISSION FROM ACLENTRIES;deleteUser2=DELETE FROM USERMEMBER WHERE USERID
= ?;getPermission=SELECT DISTINCT PERMISSION FROM ACLENTRIES WHERE
PERMISSION = ?;getUser=SELECT USERID, PASSWORD FROM WLSUSER WHERE USERID =
?;deleteGroup2=DELETE FROM ACLENTRIES WHERE PRINCIPAL =
?;deleteGroup1=DELETE FROM GROUPMEMBER WHERE GROUPID =
?;deleteUser1=DELETE FROM ACLENTRIES WHERE PRINCIPAL =
?;getAclEntries=SELECT NAME, PRINCIPAL, PERMISSION FROM ACLENTRIES WHERE
NAME = ? ORDER BY PRINCIPAL;getGroupMembersGroups=SELECT GROUPMEMBERID,
GROUPID FROM GROUPMEMBER WHERE GROUPID = ?;getGroups=SELECT GROUPID FROM
WLSGROUP ORDER BY GROUPID;getGroup=SELECT GROUPID FROM WLSGROUP WHERE
GROUPID = ?;getUsers=SELECT USERID, PASSWORD FROM WLSUSER ORDER BY
USERID"/>
```

Bold indicates values updated based on information provided during installation.

The last section of the file deploys the WebLogic internal management Web applications, and sets up the WebLogic Logic Process Integrator caching realm for the RDBMS security realm.

Figure A-11 The config.xml File: Part 8

```
<Log
   Name="config/mydomain/mydomain"/>

<Application
   Name="wl_management_internal2"
   Path="./config/tmp">
      <WebAppComponent
         Name="wl_management_internal2"
         Targets="myserver"
         URI="wl_management_internal2.war"/>
</Application>

<Application
   Name="wl_management_internal1"
   Path="./config/tmp">
      <WebAppComponent
         Name="wl_management_internal1"
         Targets="myserver"
         URI="wl_management_internal1.war"/>
</Application>

<CachingRealm
   BasicRealm="wlpiRDBMSRealm"
   CacheCaseSensitive="true"
   Name="wlpiCachingRealm"/>
</Domain>
```

setEnv.cmd/setEnv.sh

This executable file is used to set the environment variables for WebLogic Collaborate. A version of this file resides in the `bin` directory under the WebLogic Collaborate installation directory, and in each domain directory. This file is called by the `startWeblogic` command file and other command files provided with the WebLogic Collaborate distribution.

The `setEnv.cmd` (Windows) or `setEnv.sh` (UNIX) file for the `collaborate/config/mydomain` domain is shown in the following figure. The values in bold are set during installation based on your installation location.

Figure A-12 The setEnv Command

```
@REM  Set home directory location variables.
@REM ------------------------------------------------------------------------+

set JAVA_HOME=C:\bea\jdk130
set BEA_HOME=C:\bea
set WL_HOME=C:\bea\wlserver6.0
set WLC_HOME=C:\bea\wlintegration2.0\collaborate
set WLPI_HOME=C:\bea\wlintegration2.0\processintegrator
set WLINT_HOME=C:\bea\wlintegration2.0
set WLC_MYDOMAIN_HOME=%WLC_HOME%\config\mydomain
set WLC_MYDOMAIN_CLOUDSCAPE_HOME=%WLINT_HOME%\repository\cloudscape

@REM  Set WLCCLASSPATH
@REM ------------------------------------------------------------------------

set WLCCLASSPATH=%WLCCLASSPATH%;%WLC_HOME%\lib\lwclient.jar


@REM  Set WLPICLASSPATH
@REM ------------------------------------------------------------------------

set WLPICLASSPATH=%WLPICLASSPATH%;%WLPI_HOME%\lib\rdbmsrealm.jar


@REM  Set JAVACLASSPATH
@REM ------------------------------------------------------------------------

set JAVACLASSPATH=%WLPICLASSPATH%;%JAVACLASSPATH%;%WLCCLASSPATH%;


@REM  Prepend to existing PATH
@REM ------------------------------------------------------------------------

set PATH=%WLC_HOME%\bin;%PATH%
```

startWebLogic.cmd/startWeblogic.sh

This executable command file starts WebLogic Server. Using the content of the `config.xml` file, it deploys WebLogic Collaborate, WebLogic Process Integrator, and the WebLogic Collaborate plug-in for WebLogic Process Integrator. It is installed in both the `collaborate/config/samples` and `collaborate/config/mydomain` directories.

The `startWeblogic.cmd` (Windows) or `startWeblogic.sh` (UNIX) file for the `collaborate/config/mydomain` domain is shown in the following figure. The arrows indicate the continuation of a line that appears as a single line in the file.

Figure A-13 The startWeblogic Command

```
@REM  Cd to mydomain directory.
@REM ========================================================================

@REM Copy the wlc-wlpi-plugin.jar into WLPI home

copy C:\bea\wlintegration2.0\collaborate\lib\wlc-wlpi-plugin.jar
C:\bea\wlintegration2.0\processintegrator\lib

cd /d C:\bea\wlintegration2.0\collaborate\config\mydomain


@REM  Call setenv.
@REM ------------------------------------------------------------------------

call setenv.cmd


@REM   WLS should be started in the same directory where the config directory
@REM   is located.
@REM ------------------------------------------------------------------------

cd ..\..


@REM  Start weblogic
@REM ------------------------------------------------------------------------

%JAVA_HOME%\bin\java -classic -ms64m -ms64m -classpath %START_WL_CLASSPATH%
-Dbea.home=%BEA_HOME% -Dweblogic.home=%WL_HOME%
-Dweblogic.system.home=%WLC_MYDOMAIN_HOME%
-Dweblogic.Domain=mydomain
-Dweblogic.management.password=security
-Dcloudscape.system.home=%WLC_MYDOMAIN_CLOUDSCAPE_HOME%
-Dweblogic.Name=myserver
-Djava.security.policy=%WL_HOME%\lib\weblogic.policy weblogic.Server


@REM  The following argument can be passed to java above to enable all debugging info:
@REM -Dbea.eci.debug=X1-89
```

The Password is provided with this statement. If you change the password you must update accordingly.

The `-Dweblogic.management.password=security` option in the `java` command that starts WebLogic Server provides the password. If this option is removed, you will be prompted for a password. If you change the password, you must remove or update this option.

fileRealm.properties

This properties file controls the User, Group, and ACL objects that are created when WebLogic Server is started.

The `fileRealm.properties` file for the `collaborate/config/mydomain` domain is shown in the following figure.

Figure A-14 The fileRealm.propertis File: Part 1

```
#Wed Jun 20 19:01:40 PDT 2001
acl.reset.weblogic.jdbc.connectionPool.wliPool=admin,guest
acl.reset.weblogic.jdbc.connectionPool.wlpiPool=admin,guest
user.admin=0xab3a488db0652704287970cdf97854812feea77b
group.CreateTemplate=admin,joe,mary,guest,wlcsystem,wlpisystem
acl.list.weblogic.jndi.weblogic=system,everyone
acl.lockServer.weblogic.admin=system
acl.list.weblogic.jndi.weblogic.ejb=system,everyone
group.wlpiUsers=admin,joe,system,mary,wlcsystem,guest,wlpisystem
acl.list.weblogic.jndi.weblogic.rmi=everyone
acl.reserve.weblogic.jdbc.connectionPool.wliPool=everyone
acl.execute.weblogic.servlet.AdminThreads=system
acl.hubconfig.WLCAdmin=admin
acl.write.weblogic.workspace=system,everyone
user.mary=0xa078cb45e6f6c4eefdd1f14495ff739b5536904c
acl.shrink.weblogic.jdbc.connectionPool.wlpiPool=admin,guest
user.wlpisystem=0xfd511836a8e67fa799348c8635de15185677aff2
acl.execute.weblogic.servlet.T3AdminMain=system
acl.unlockServer.weblogic.admin=system
group.ConfigureComponents=admin,joe,mary,guest,wlcsystem,wlpisystem
acl.execute.weblogic.servlet.AdminJDBC=system
acl.shrink.weblogic.jdbc.connectionPool.examplesPool=everyone
acl.execute.weblogic.servlet.AdminRealm=system
user.wlcsystem=0xab3a488db0652704287970cdf97854812feea77b
acl.modify.weblogic.jndi.weblogic=system,everyone,guest
acl.execute.weblogic.servlet.ConsoleHelp=everyone
acl.execute.weblogic.servlet.Certificate=system
group.wlcSamplesGroup=guest,wlpisystem
acl.modify.weblogic.jndi.weblogic.ejb=system,everyone,guest
acl.reset.weblogic.jdbc.connectionPool.examplesPool=admin,guest
acl.modify.weblogic.jndi.weblogic.rmi=everyone
acl.execute.weblogic.servlet.classes=everyone
group.ExecuteTemplate=admin,joe,mary,guest,wlcsystem,wlpisystem
acl.execute.weblogic.servlet=system,everyone
acl.modify.weblogic.admin.acl=system
acl.lookup.weblogic.jndi.weblogic.ejb=system,everyone,guest
acl.execute.weblogic.servlet.AdminClients=system
acl.read.managedObject=system
```

Figure A-15 The fileRealm.propertis File: Part 2

```
group.wlpiAdministrators=admin,wlpisystem
acl.lookup.weblogic.jndi.weblogic.rmi=everyone
acl.lookup.weblogic.admin.mbean.MBeanHome=everyone
acl.list.weblogic.jndi.weblogic.fileSystem=everyone
group.Role2Org2=admin,joe,mary
group.Role2Org1=admin,joe,mary
group.ConfigureSystem=admin,joe,mary,guest,wlcsystem,wlpisystem
acl.lookup.weblogic.jndi.weblogic.fileSystem=everyone
acl.enablermonitor.WLCAdmin=admin
user.joe=0xa078cb45e6f6c4eefdd1f14495ff739b5536904c
group.DeleteTemplate=admin,joe,mary,guest,wlcsystem,wlpisystem
group.AdministerUser=admin,joe,mary,guest,wlcsystem,wlpisystem
acl.create.weblogic.jms.ServerSessionPool=everyone
acl.execute.weblogic.servlet.AdminEvents=system
group.UpdateTemplate=admin,joe,mary,guest,wlcsystem,wlpisystem
acl.execute.weblogic.servlet.AdminVersion=system
group.adminGroup=admin,joe,mary
group.Role1Org2=admin,joe,mary
group.Role1Org1=admin,joe,mary
acl.shrink.weblogic.jdbc.connectionPool.wliPool=admin,guest
group.MonitorInstance=admin,joe,mary,guest,wlcsystem,wlpisystem
acl.execute.weblogic.servlet.containerManaged=everyone
user.system=0xab3a488db0652704287970cdf97854812feea77b
acl.write.managedObject=system
acl.execute.weblogic.servlet.AdminMain=system
acl.execute.weblogic.servlet.AdminLicense=system
acl.execute.weblogic.servlet.AdminProps=system
acl.modify.weblogic.jndi.weblogic.fileSystem=everyone
acl.lookup.weblogic.jndi.weblogic=everyone
acl.read.weblogic.workspace.system=system,everyone
acl.shutdown.weblogic.admin=system
acl.reserve.weblogic.jdbc.connectionPool=system
acl.write.weblogic.workspace.system=system,everyone
acl.execute.weblogic.servlet.AdminConnections=system
acl.reset.weblogic.jdbc.connectionPool=system
acl.read.weblogic.workspace=system,everyone
acl.boot.weblogic.server=system,everyone
acl.access.weblogic.admin.mbean.MBeanHome=everyone
```

B Update Considerations

Table B-1 summarizes the constraints and considerations related to modifying the configuration for WebLogic Collaborate, trading partners, conversation definitions, collaboration agreements, logic plug-ins, and business protocols.

Table B-1 WebLogic Collaborate Update Considerations

Element	Category	Parameter	Update Considerations
WebLogic Collaborate (WLC)	General	WLC Server 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	

Table B-1 WebLogic Collaborate Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
WebLogic Collaborate (WLC) (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 on the navigation tree	

Table B-1 WebLogic Collaborate Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Trading Partners	General	Name	Cannot be updated. To update a trading partner name, delete the trading partner and recreate it with the new name. **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	Can be added at any time. An existing party ID cannot be updated if the trading partner is engaged in a trading partner session.
		Business ID	Cannot be updated if the trading partner is engaged in a trading partner session.
		Business ID Type	
	Certificates	Certificate Name	A 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.
		Certificate Type	
		Certificate Location	
		Private Key Location	

Table B-1 WebLogic Collaborate 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. An existing document exchange cannot be updated while WebLogic Collaborate is up.
		Business Protocol Binding	
		Business Protocol Definition	
		End Point Type	
	Doc Exchange XOCP Specific Settings	Confirmed Delivery	Cannot be updated while WebLogic Collaborate is up.
		Message History	
		Retries	
		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.
	Doc Exchange cXML Specific Settings	Shared Secret	Cannot be updated while WebLogic Collaborate is up.
		Signature Certificate	
	Doc Exchange RosettaNet 1.1 Specific Settings	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 B-1 WebLogic Collaborate Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Trading Partners (Continued)	Transport	Transport Name	A transport can be added at any time. Cannot be updated while WebLogic Collaborate is up.
		Transport Protocol	
		Security Protocol	
		Endpoints	
	Delivery Channels	Delivery Channel Name	Cannot be updated while WebLogic Collaborate is up.
		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. To update a conversation definition name, delete the definition and recreate it with the new name. **Note:** You cannot delete a conversation definition if an active conversation uses 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	

Table B-1 WebLogic Collaborate Update Considerations (Continued)

Element	Category	Parameter	Update Considerations
Collaboration Agreements	General	Collaboration Agreement Name	Cannot be updated. To update a collaboration agreement name, delete the agreement and recreate it with the new name. **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 is registered.
		Conversation Definition	
	Parties	Trading Partner	An existing party definition cannot be updated while WebLogic Collaborate is up. Parties can be added at any time.
		Role	
Logic Plug-Ins	General	Name	Cannot be updated. To update a logic plug-in name, delete the logic plug-in and recreate it with the new name. **Note:** When you delete a logic plug-in, the plug-in remains in use until WebLogic Server is shut down and restarted.
		Description	Custom logic plug-in definitions can be added, 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.
		Type	
		Java Class Name	
		Parameters for Java Class	

Table B-1 WebLogic Collaborate 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	