BEA WebLogic Collaborate

C-Hub Administration Guide

BEA WebLogic Collaborate 1.0.1
Document Edition 1.0.1
March 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA WebLogic Collaborate C-Hub Administration Guide

Document Edition	Date	Software Version
1.0.1	March 2001	1.0.1

Contents

2. Setting Up the C-Hub

3. Working with the Bulk Loader

4. Configuring Security

Part II. Using the C-Hub Administration Console

8. Getting Started with C-Hub Administration

9. Creating and Modifying C-Hubs

10. Configuring Document Definitions for a C-Hub

11. Configuring Message Definitions for a C-Hub

12. Working with Logic Plug-Ins

13. Working with Business Protocol Definitions

14. Working with Trading Partners

15. Setting Up Conversations

16. Working with C-Spaces

17. Setting Preferences

18. Monitoring the C-Hub

Index

About This Document

This document describes how to configure and manage c-hubs in the BEA WebLogic Collaborate™ system.

The document is organized as follows:

- Chapter 1, “Introducing the C-Hub,” introduces c-hubs and the key concepts for configuring and monitoring a c-hub.
- Chapter 2, “Setting Up the C-Hub,” describes how to configure a c-hub.
- Chapter 3, “Working with the Bulk Loader,” describes how to use the Bulk Loader to import, export, and delete repository data.
- Chapter 4, “Configuring Security,” describes how to configure security for applications and provides conceptual information about WebLogic Collaborate security.
- Chapter 5, “Configuring Persistence and Recovery,”describes how to configure persistence and recovery and provides conceptual information about these topics.
- Chapter 6, “Configuring Business Protocols,” describes how to configure and work with business protocols.
- Chapter 7, “Routing and Filtering XOCP Business Messages,” describes how to configure XPath routers and XPath filters and provides conceptual information about routers and filters.
- Chapter 8, “Getting Started with C-Hub Administration,” describes how to get started with c-hub administration.
- Chapter 9, “Creating and Modifying C-Hubs,” describes how to create and modify c-hubs.
- Chapter 10, “Configuring Document Definitions for a C-Hub,” describes how to configure document definitions for a c-hub.

- Chapter 11, “Configuring Message Definitions for a C-Hub,” describes how to configure message definitions for a c-hub.
- Chapter 12, “Working with Logic Plug-Ins,” describes how to work with logic plug-ins.
- Chapter 13, “Working with Business Protocol Definitions,” describes how to work with business protocol definitions.
- Chapter 14, “Working with Trading Partners,” describes how to work with trading partners.
- Chapter 15, “Setting Up Conversations,” describes how to set up conversations.
- Chapter 16, “Working with C-Spaces,” describes how to work with c-spaces.
- Chapter 17, “Setting Preferences,” describes how to set preferences.
- Chapter 18, “Monitoring the C-Hub,” describes how to monitor a c-hub.

What You Need to Know

This document is intended mainly for:

- E-market owners who create e-markets that run on the BEA WebLogic Collaborate system.
- System administrators who set up and administer c-hubs and c-spaces.
- Application developers who develop external management applications to monitor c-hubs at runtime.

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

e-docs Web Site

The BEA WebLogic Collaborate product documentation is available on the BEA e-docs Web site at `http://e-docs.bea.com`.

How to Print this Document

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

A PDF version of this document is available 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

For more information about Java 2 Enterprise Edition (J2EE), eXtended Markup Language (XML), and Java programming, see the *Bibliography* in the BEA WebLogic Collaborate online documentation.

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 BEA WebLogic Collaborate documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Collaborate 1.0.1 release.

If you have questions about this version of BEA WebLogic Collaborate, or if you have problems installing and running BEA WebLogic Collaborate, contact BEA Customer Support at `http://www.bea.com/support`. 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
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 filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

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

Convention	Item

`[ ]`	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.

Part I Configuring and Running the C-Hub

CHAPTER

1 Introducing the C-Hub

The following sections introduce the c-hub component of the BEA WebLogic Collaborate system:

- C-Hubs
- Architecture
- Message Processing
- Configuration and Monitoring
- Repository
- Responsibilities and Tasks

Before you read this document, it is recommended that you read the *BEA WebLogic Collaborate Getting Started* document.

C-Hubs

The following sections describe c-hubs:

- C-Hubs and C-Enablers
- C-Hub Services

C-Hubs and C-Enablers

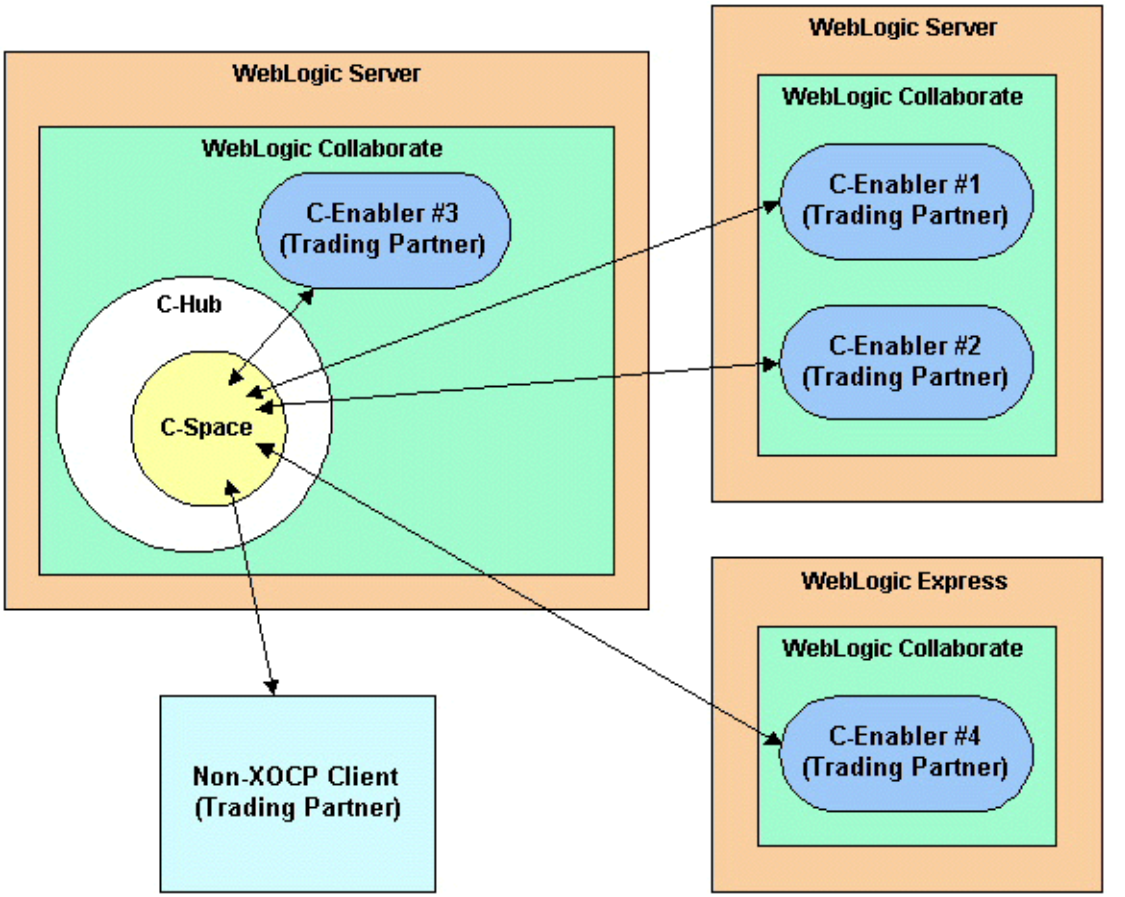
An electronic marketplace, or e-market, is an environment through which companies conduct business-to-business e-commerce. E-markets offer a set of services to conduct, manage, and orchestrate conversations between various trading partners.

WebLogic Collaborate offers a flexible, extendable, and scalable solution for implementing e-markets. A WebLogic Collaborate system consists of the following main parts:

- A c-hub
- One or more c-enablers

The following figure shows an example of the relationships among c-hubs, c-spaces, and c-enablers. Each trading partner uses a c-enabler to join a c-space, which exists in a c-hub. By joining a c-space, the trading partner can participate in conversations with other trading partners.

Figure 1-1 Architecture of a Possible WebLogic Collaborate Configuration



In the figure, c-enabler #3 is colocated with the c-hub, which means that they are deployed on the same BEA WebLogic Server™ instance. A c-hub runs on a single WebLogic Server instance, and a WebLogic Server instance can host a single c-hub. However, multiple c-enablers can run on a single WebLogic Server instance, as illustrated by c-enablers #1 and #2 in the figure. Instead of running on WebLogic Server, a c-enabler can run on WebLogic Express as illustrated by c-enabler #4.

A c-hub node (the machine on which a c-hub is running) can host one or more c-enablers, as illustrated by c-enabler #3. In the figure, the c-enablers are deployed on various WebLogic Servers or WebLogic Express, but all the c-enablers participate in the same c-space. A non-XOCP client can also participate in a c-space.

A c-hub represents an e-market owner and can host multiple c-spaces. Each trading partner can subscribe to business conversations, which are communicated through a mediated messaging system, and define local workflow and actions to execute upon receipt of business events (messages or documents).

The c-enabler is a 100-percent Java class library that is deployed at each participant node to allow access to c-spaces hosted on a c-hub. The c-enabler is typically downloaded with authorization from the e-market administrator. For a detailed description of c-enablers, see the *BEA WebLogic Collaborate C-Enabler Administration Guide*.

For a complete explanation of e-markets, collaboration, and other related concepts, see the Overview section in *BEA WebLogic Collaborate Getting Started*.

C-Hub Services

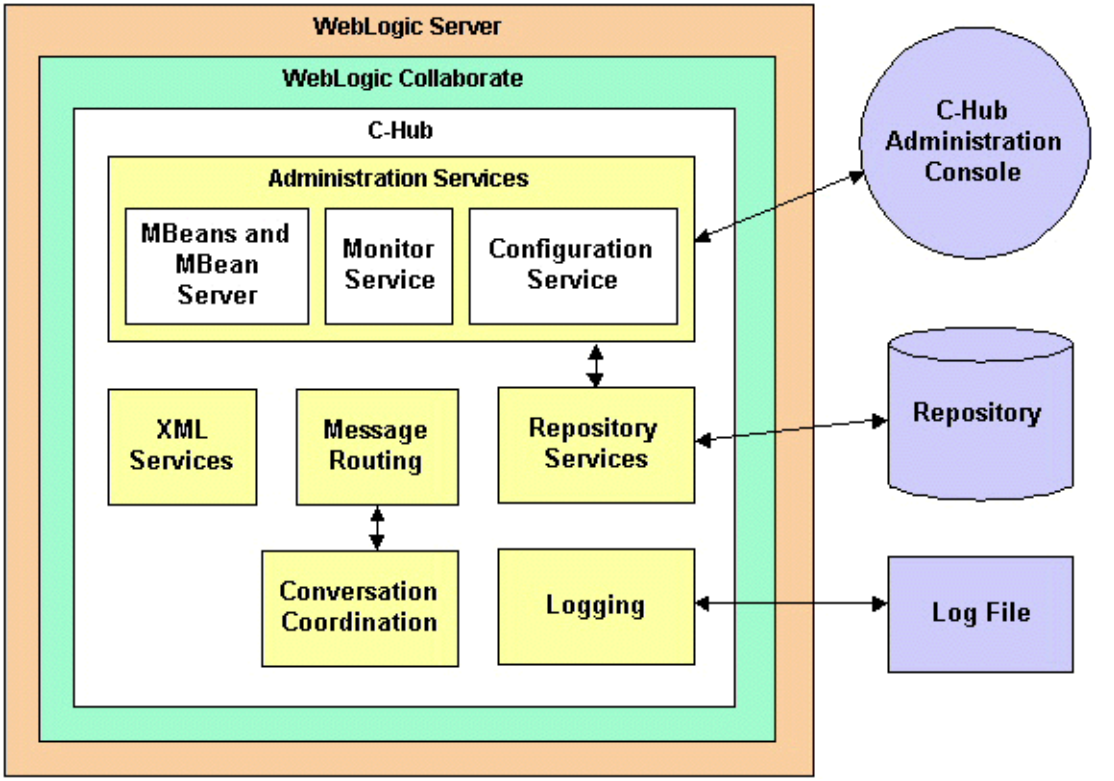
A c-hub provides the following shared services to c-enablers:

- Subscription management
- Conversation coordination
- Security
- Administration
- Routing and filtering
- Logging
- XML services

Architecture

The following figure shows the c-hub architecture. WebLogic Collaborate runs on WebLogic Server. The c-hub can integrate with any packaged, custom, or mainframe applications running on WebLogic Server.

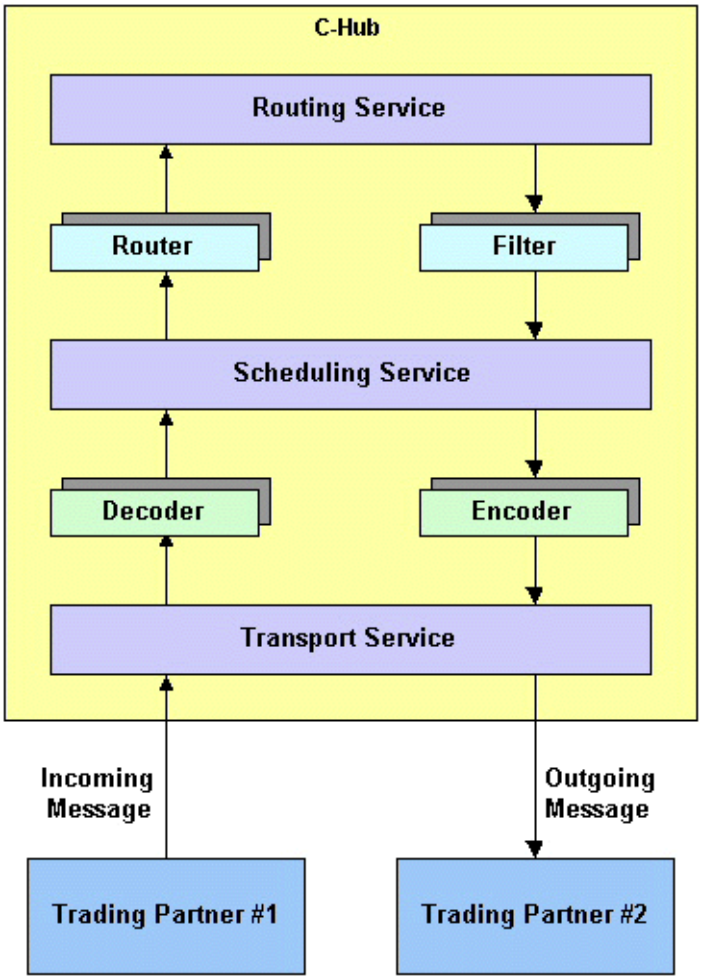
Figure 1-2 C-Hub Architecture



Message Processing

The following figure illustrates how a c-hub processes a message.

Figure 1-3 Message Processing



The following steps describe the preceding figure:

1. The **transport service** reads the incoming message and forwards it to the appropriate decoder based on the message protocol, such as XOCP or RosettaNet. The URL on which the transport service receives the message identifies the protocol and the c-space. For information about business protocols, see Chapter 6, "Configuring Business Protocols."

2. The **decoder** processes the protocol-specific headers, identifies the sending trading partner, enlists the sending trading partner in a conversation, prepares a reply to return to the sender, and forwards the message to the scheduling service.

3. The **scheduling service** enqueues the message to store it for subsequent retrieval and forwards the message to the router.

4. The **router** determines the trading partners to whom the message should be routed and forwards the message to the routing service. You can use logic plug-ins to add recipients to or delete recipients from the list. For information about routing, see Chapter 7, "Routing and Filtering XOCP Business Messages." For information about logic plug-ins, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*.

5. The **routing service** performs the final validation of the message recipients, stores the message for delivery to each validated recipient, and forwards one copy of the message to the filter for each validated recipient.

6. The **filter** receives the message for a specific recipient and determines whether or not to send the message to the recipient. You can use logic plug-ins to affect how this decision is made. If the message is going to be sent to the recipient, the filter forwards the message to the scheduling service. For information about filtering, see Chapter 7, "Routing and Filtering XOCP Business Messages." For information about logic plug-ins, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*.

7. The **scheduling service** performs additional internal operations related to quality of service issues and conversation management. The scheduling service forwards the message to the encoder. For information about quality of service, see the *BEA WebLogic Collaborate Developer Guide*.

8. The **encoder** transforms the message as necessary to support the business protocol and forwards the message to the transport service.

9. The **transport service** sends the message to the recipient.

Configuration and Monitoring

The C-Hub Administration Console enables you to create, configure, and monitor a c-hub. For information about the C-Hub Administration Console, see "Using the C-Hub Administration Console," which is Part II in this document.

The Bulk Loader enables you to create and configure a c-hub. For information about the Bulk Loader, see Chapter 3, "Working with the Bulk Loader."

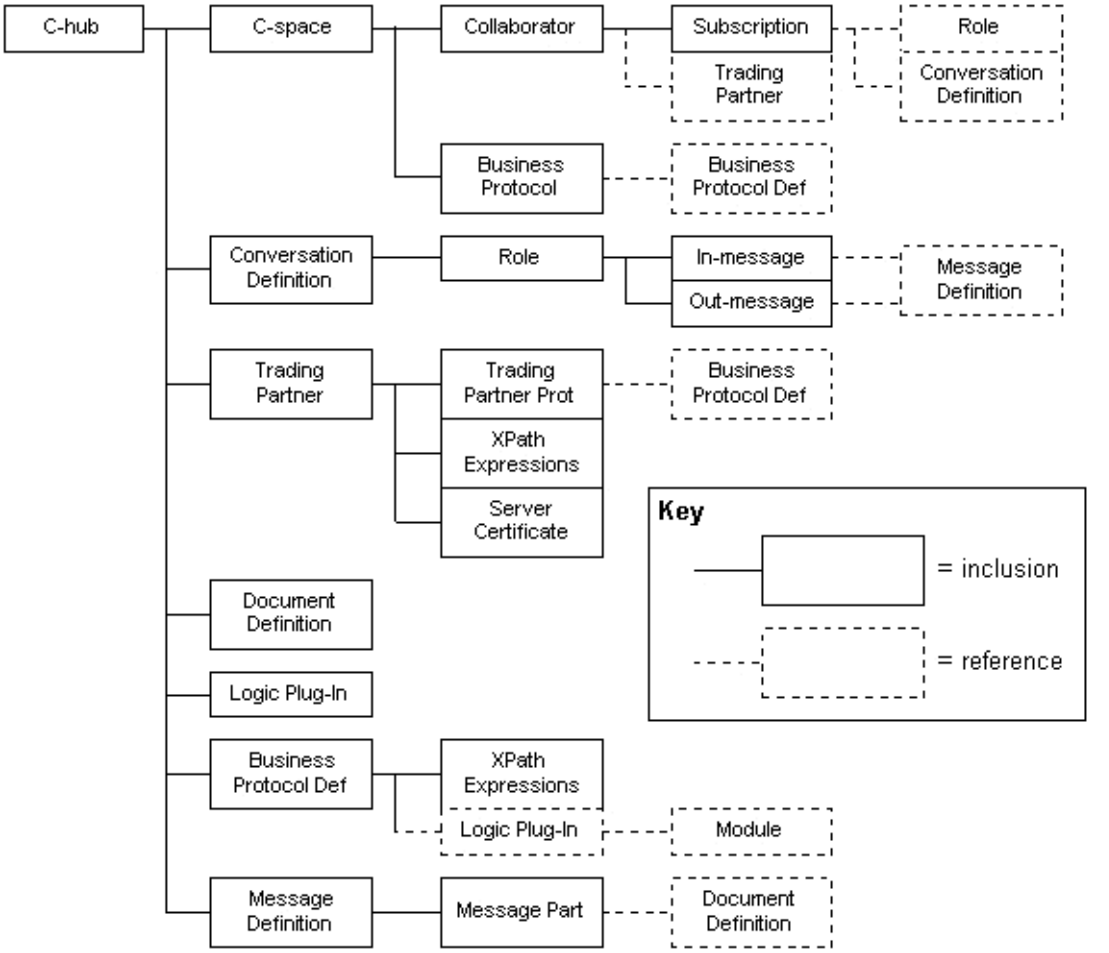
You can create c-hub management applications that use the c-hub MBeans to monitor c-hub run-time activities. MBeans function as a monitoring API that is based on the Java Management Extensions (JMX) published by Sun Microsystems, Inc. For information about working with the c-hub MBeans, see the *BEA WebLogic Collaborate Developer Guide*.

Repository

The repository is a database that stores configuration information for the c-hub object. The following figure shows the relationships among elements in the repository. Solid lines represent inclusion, which means that if you remove an element, all the inclusion elements are also deleted. For example, if you remove a c-space, then all the collaborators for that c-space are also removed. Dashed lines represent reference, which means that if you remove an element, the elements referenced by the removed

element are not removed. For example, if you remove a business protocol, the reference to the business protocol definition is removed, but the business protocol definition remains.

Figure 1-4 Element Relationships



To manage the information in the repository, you can:

- Use the C-Hub Administration Console as described in "Using the C-Hub Administration Console," which is Part II in this document.
- Use the Bulk Loader as described in Chapter 3, "Working with the Bulk Loader."

The following sections provide overviews of the main elements in the repository:

- C-Hubs
- C-Spaces
- Conversation Definitions and Roles
- Document Definitions
- Trading Partners and Trading Partner Protocols
- Logic Plug-Ins
- Business Protocols and Business Protocol Definitions
- Message Definitions

C-Hubs

Description:	The c-hub is the root definition in the repository. Most high-level elements such as c-spaces, conversation definitions, trading partners, document definitions, logic plug-ins, and business protocol definitions stem from this root.
Relationships:	There is a one-to-many relationship between a c-hub and the other elements in the repository. For example, one c-hub can host multiple c-spaces; but for every c-space, there is only one c-hub. Likewise, one c-hub can host multiple trading partners; but for every trading partner, there is only one c-hub.

Interface: C-hubs are represented by the `HubMBean` interface in the `com.bea.b2b.management.hub.runtime` package. C-hub management applications use `HubMBean` objects to monitor c-hubs.

C-Spaces

Description: A c-space is a virtual place where predefined trading partners can conduct and coordinate conversations with each other. A c-space has a specific business purpose and logically represents an electronic marketplace or common trading environment. It provides the administration capability, conversation coordination, and underlying messaging services that are used to create a dynamic business-to-business integration environment.

Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol.

Relationships: There is a one-to-many relationship between a c-hub and c-spaces. A c-hub can host multiple c-spaces; but for every c-space, there is only one c-hub.

There is a one-to-many relationship between a c-space and business protocols. There is also a one-to-many relationship between a c-space and collaborators. For example, one c-space can include many business protocols and many collaborators.

Interface: C-spaces are represented by the `CSpaceMBean` interface in the `com.bea.b2b.management.hub.runtime` package. C-hub management applications use `CSpaceMBean` objects to monitor c-spaces.

Conversation Definitions and Roles

Description:	A conversation is a series of predefined message exchanges between trading partners that take place in a c-space in the context of a predefined business model. Each message in the conversation can cause any number of back-end transactions. A conversation definition is a set of roles pertaining to one conversation. A role is the subscription unit in a conversation. To participate in a conversation, a trading partner subscribes to a specific role that is defined in the conversation definition associated with the conversation. A role is defined by a sequence of documents that can be sent or received by a trading partner in the conversation. The role defines what a trading partner can do, such as buy or sell. Each conversation definition has two or more roles.
Relationships:	There is a one-to-many relationship between a c-hub and conversation definitions. A c-hub has multiple conversation definitions; but for every conversation definition, there is only one c-hub. A trading partner uses a subscription to relate a conversation definition to a c-space. There is a one-to-many relationship between a conversation definition and roles. A conversation definition has multiple roles; but for every role in the conversation, there is only one conversation definition.
Interface:	Conversation definitions are represented by the `GlobalConversationMBean` interface in the `com.bea.b2b.management.hub.runtime` package. C-hub management applications use `GlobalConversationMBean` objects to monitor global conversations.

Document Definitions

Description:	A document definition is a schema (such as a DTD) that defines a valid document.
Relationships:	There is a one-to-many relationship between a c-hub and document definitions. A c-hub has multiple document definitions; but for every document definition, there is only one c-hub. There is a many-to-many relationship between document definitions and message definitions. For example, a document definition can be used in many message definitions. And a message definition can be referenced by many document definitions.
Interface:	None.

Trading Partners and Trading Partner Protocols

Description: A trading partner is a representation of an object, such as a company, that is authorized to participate in one or more c-spaces. A trading partner needs a separate authorization for each c-space that it participates in. A trading partner uses the c-enabler to subscribe to conversations that are communicated through a mediated messaging system on the c-hub.

A trading partner protocol defines additional information that might be needed by the protocol that a trading partner uses. Trading partner protocols are required only for non-XOCP business protocols. For information about configuring non-XOCP business protocols, see Chapter 6, “Configuring Business Protocols.”

Relationships: To participate in conversations in a c-space, a trading partner must register as a collaborator on a c-hub. C-hub administrators define and manage collaborator registration. Before registering on a c-hub, a trading partner must first establish a business relationship with the e-market owner.

There is a one-to-many relationship between a c-hub and trading partners. A c-hub has multiple trading partners; but for every trading partner, there is only one c-hub.

There is a many-to-many relationship between trading partners and c-spaces, and between trading partners and conversation definitions. For example, multiple trading partners can join one c-space, and one trading partner can join multiple c-spaces on the c-hub. A trading partner needs a separate c-enabler for each c-space that the partner joins.

Trading partners can participate in multiple conversations within a c-space. To participate in a conversation, the trading partner must subscribe to a role in a conversation definition.

To participate in a non-XOCP conversation, a trading partner might need to define a trading partner protocol. There is a one-to-many relationship between a trading partner and trading partner protocols. There is a many-to-one relationship between trading partner protocols and a business protocol definition.

For example, one trading partner can define several different trading partner protocols for participating in several different types of conversations. Each trading partner protocol must reference one business protocol definition, and each business protocol definition can be referenced by multiple trading partner protocols.

Interface: Trading partners are represented by the `CollaboratorMBean` interface in the `com.bea.b2b.management.hub.runtime` package. C-hub management applications use `CollaboratorMBean` objects to monitor trading partners.

Logic Plug-Ins

Description:	A logic plug-in is a component that provides additional processing for information that passes through the c-hub. For information about logic plug-ins, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*.
Relationships:	There is a one-to-many relationship between a c-hub and logic plug-ins. A c-hub can have many logic plug-ins; but for every logic plug-in, there is only one c-hub.
Interface:	None.

Business Protocols and Business Protocol Definitions

Description:	A business protocol specifies a URL and refers to a business protocol definition. Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol. A business protocol definition specifies how the c-hub processes business messages, including how to read the messages and how to route the messages to the recipients. A business protocol definition also specifies persistence, retries, and quality of service. For information about business protocols, see Chapter 6, "Configuring Business Protocols."
Relationships:	A c-space has a one-to-many relationship with business protocols, and a c-hub has a one-to-many relationship with business protocol definitions. Business protocol definitions have a many-to-one relationship with a business protocol. For example, a c-space can include many business protocols and a c-hub can include many business protocol definitions. Each business protocol can reference one business protocol definition. And each business protocol definition can be referenced by many business protocols.
Interface:	None.

Message Definitions

Description:	A message definition defines the business content (business documents and attachments) of a business message. A message definition consists of ordered message parts: ■ XML-type message parts, which are business documents and require document definitions. ■ Binary-type message parts, which are attachments and require no other information.
Relationships:	A c-hub has a one-to-many relationship with message definitions. Roles have a many-to-many relationship with message definitions. For example, a c-hub can include many message definitions. Each role includes messages that refer to message definitions. A message definition can be associated with roles in multiple conversation definitions.
Interface:	None.

Responsibilities and Tasks

The following responsibilities are associated with c-hubs:

- **E-Market owners** design c-spaces and negotiate with trading partners to participate.
- **Flow designers** design business processes, workflows, and rules; and define the associated conversations, roles, and documents.
- **Business developers** configure the elements in the repository.
- **System administrators** install, set up, and monitor c-hubs and c-spaces; and monitor log files at run time.

- **Application developers** develop applications that run in the WebLogic Collaborate system, including external management applications to monitor c-hubs at run time.

Working with a c-hub involves the following tasks:

- Installing the WebLogic Collaborate software according to the instructions in the *BEA WebLogic Collaborate Installation Guide*.
- Configuring a c-hub according to the instructions in Chapter 2, "Setting Up the C-Hub."
- Populating the repository with configuration information using the C-Hub Administration Console as described in "Using the C-Hub Administration Console," which is Part II in this document. Populating the repository includes the following tasks:
  - Creating and configuring the c-hub
  - Creating and configuring one or more c-spaces
  - Adding trading partners and defining trading partner protocols
  - Adding business protocols and business protocol definitions
  - Adding document definitions, message definitions, and conversation definitions
  - Defining subscriptions
- Monitoring the c-hub at run time using one of the following:
  - The C-Hub Administration Console as described in Chapter 18, "Monitoring the C-Hub."
  - Management applications that retrieve c-hub run-time information by using the c-hub MBeans and MBean server. For more information, see the *BEA WebLogic Collaborate Developer Guide*.

CHAPTER

2 Setting Up the C-Hub

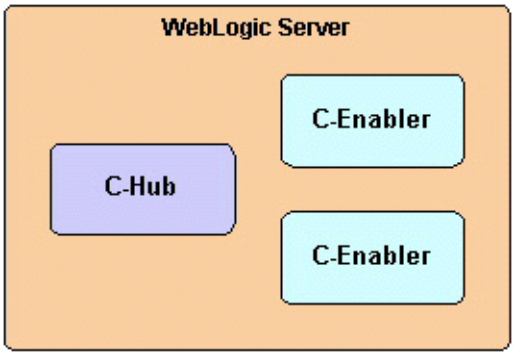
The following sections describe how to set up a c-hub:

- About the C-Hub Administration Console
- Configuring the C-Hub Startup Class and Starting the C-Hub
- Configuring the Repository
- Configuring the Java Message Service Queue
- Configuring and Running the C-Hub Administration Console
- Sample C-Hub Sections in the config.xml File

About the C-Hub Administration Console

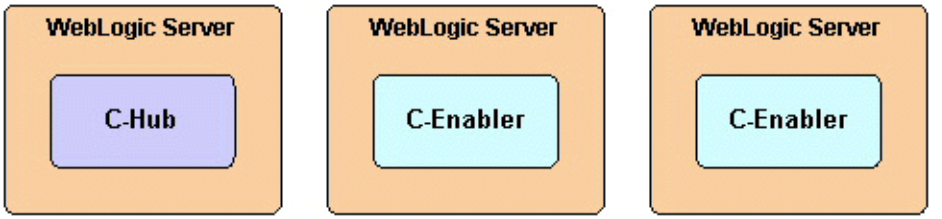
Many configuration tasks require changes to the `config.xml` file. The number of configuration changes that you need to make depends on whether or not the c-hub and c-enablers are colocated. The following figure shows a c-hub and c-enablers that are colocated, which means that they run on the same WebLogic Server instance.

Figure 2-1 C-Hub and C-Enablers That Are Colocated



The following figure shows a c-hub and c-enablers that are not colocated, which means that they run on different WebLogic Servers.

Figure 2-2 C-Hub and C-Enablers That Are Not Colocated



If the c-hub and c-enablers are colocated, they use the same configuration. If the c-hub and c-enablers are not colocated, they use different configurations—one for each WebLogic Server instance.

The `config.xml` files are in the following locations:

- **For the c-hub**: in the `hub/config/mydomain` subdirectory of your WebLogic Collaborate installation directory. WebLogic Collaborate uses this `config.xml` file for the c-hub and for any colocated c-enablers.
- **For the c-enabler**: in the `enabler/config/mydomain` subdirectory of your WebLogic Collaborate installation directory.

When you run the Verification example to verify the WebLogic Collaborate installation as described in the *BEA WebLogic Collaborate Installation Guide*, the example software sets up the `config.xml` file for your environment. For example, you might need to use the Weblogic Server Administration Console to:

- Set values for your applications.
- Change JDBC and database values if you are using a different database for the repository than the one that was set up during installation.
- Change permission values.

This "Setting Up the C-Hub" section describes how to configure the c-hub. For information about configuring the c-enablers, see Configuring C-Enablers in the *BEA WebLogic Collaborate C-Enabler Administration Guide*. For information about configuring the Web Logic Server, see the *BEA WebLogic Server Administration Guide*.

Configuring the C-Hub Startup Class and Starting the C-Hub

There are two ways to start the c-hub, from the command line using startup scripts and from the C-Hub Administration Console.

Using the Command Line

To start the C-Hub from the command line, you must first use the WebLogic Server Administration Console and configure the C-Hub start-up class (`com.bea.b2b.hub.Startup`) to use the values shown in the following table.

Attribute	Value
Name	`WLCStartup`
Class Name	`com.bea.b2b.hub.Startup`
Arguments	`RecoveryMode=ON|OFF`
Abort Startup on Failure	`TRUE`

For more information on registering startup classes, see "Starting and Stopping WebLogic Servers" in the *BEA WebLogic Server Administration Guide*.

Using the C-Hub Administration Console

Start the c-hub using the C-Hub Administration Console. For complete information, see “Starting the C-Hub” on page 8-5 in Chapter 8, “Getting Started with C-Hub Administration.”

Configuring the Repository

The repository is a relational database that contains configuration information for c-hubs, c-spaces, trading partners, conversation definitions, document definitions, DTDs, XML documents, XML schemas, and message definitions. The c-hub uses a JDBC connection pool to access the repository.

For information about the supported database versions, see the *BEA WebLogic Collaborate Release Notes*.

To configure the repository, use the WebLogic Server Administration Console.

1. Open the WebLogic Server Administration Console. To configure the JDBC connection pool, select JDBC, Connection Pools.

   Note: All the WebLogic Collaborate components, such as the JMS queue, the repository, persistent storage, and administration, must use the same database and same JDBC connection pool.

   When you ran the Verification example, as described in the *BEA WebLogic Collaborate Installation Guide*, the example software configured a JDBC connection pool for the database that you specified during the installation procedure. If you are using the same database, then you do not need to perform this step. However, to use a different database you need to reconfigure the JDBC connection pool.

   a. Set the JDBC connection pool parameters.

      The following table describes the database-dependent JDBC connection pool parameters.

Table 2-1 Database-Dependent JBDC Connection Pool Parameters

Parameter	Description
`URL`	URL for the JDBC driver. For Cloudscape: `jdbc:cloudscape:wlcdb` For Oracle: `jdbc:oracle:thin:@<HOSTNAME>:<PORT>:<ORACLE_SERVICENAME>` For SQL Server: `jdbc:weblogic:mssqlserver4`
`DriverName`	Package name of the JDBC driver. For Cloudscape: `COM.cloudscape.core.JDBCDriver` For Oracle: `oracle.jdbc.driver.OracleDriver` For SQL Server: `weblogic.jdbc.mssqlserver4.Driver`
`Properties`	User name, password, and database name. For Cloudscape: (These values are ignored.) For Oracle: `user=<ORACLE_USER>;password=<ORACLE_PASSWORD>` For SQL Server: `user=<MSSQL_USER>;password=<MSSQL_PASSWORD>; server=WLCDB@<MSSQL_HOSTNAME>`

The following listing shows the settings for a JDBC connection pool for a Cloudscape database as they would appear in the `config.xml` file. You enter these settings in the WebLogic Server Administration Console. For more information about using the WebLogic Server Administration Console, see "Overview of WebLogic Server Management" in the *BEA WebLogic Server Administration Guide*.

Listing 2-1 JDBC Connection Pool Configuration for Cloudscape

```
<!-- Cloudscape settings -->
    <JDBCConnectionPool
      CapacityIncrement="1"
      DriverName="COM.cloudscape.core.JDBCDriver"
      InitialCapacity="1"
      LoginDelaySeconds="1"
      MaxCapacity="10"
      Name="wlcPool"
      Properties="user=none;password=none"
      RefreshMinutes="0"
```

```
      ShrinkPeriodMinutes="15"
      ShrinkingEnabled="true"
      Targets="myserver"
      URL="jdbc:cloudscape:wlcdb"
    />
```

The following listing is an example of the configuration settings for a JDBC connection pool for an Oracle database entered using the WebLogic Server Administration Console.

Listing 2-2 JDBC Connection Pool Configuration for Oracle

```
 <!-- Oracle settings -->
    <JDBCConnectionPool
      CapacityIncrement="1"
      DriverName="oracle.jdbc.driver.OracleDriver"
      InitialCapacity="1"
      LoginDelaySeconds="1"
      MaxCapacity="10"
      Name="wlcPool"
      Properties="user=scott;password=tiger"
      RefreshMinutes="0"
      ShrinkPeriodMinutes="15"
      ShrinkingEnabled="true"
      Targets="myserver"
      URL="jdbc:oracle:thin:@fushigi.beasys.com:1521:fush816"
/>
```

The following listing shows the configuration settings for a JDBC connection pool for a SQL Server database. These settings are entered in the WebLogic Server Administration Console.

Listing 2-3 JDBC Connection Pool Configuration for SQL Server

```
<!-- SQL Server settings (MSSQL) -->
    <JDBCConnectionPool
      CapacityIncrement="1"
      DriverName="weblogic.jdbc.mssqlserver4.Driver"
      InitialCapacity="1"
```

```
    LoginDelaySeconds="1"
    MaxCapacity="10"
    Name="wlcPool"
    Properties="user=sa;password="
    RefreshMinutes="0"
    ShrinkPeriodMinutes="15"
    ShrinkingEnabled="true"
    Targets="myserver"
    URL="jdbc:weblogic:mssqlserver4"
  />
```

b. Set the access control list (ACL) for the JDBC connection pool. The ACL entries are in the `fileRealm.properties` file.

   For example:

```
# ACL for JDBC Connection Pool
acl.reset.weblogic.jdbc.connectionPool.wlcPool=everyone
acl.reserve.weblogic.jdbc.connectionPool.wlcPool=everyone
acl.shrink.weblogic.jdbc.connectionPool.wlcPool=everyone
```

c. Specify the data source.

   Set the JDBC data source (`WLCHub.DS`) to the name of the JDBC connection pool. For example:

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

   For more information about configuring a JDBC connection pool, see "Managing JDBC Connectivity" in the *BEA WebLogic Server Administration Guide* and "JDBC Administration through the Administration Console" in the Managing Transactions section of the *BEA WebLogic Server Administration Guide*.

Warning: After you modify the `config.xml` file, you must stop and restart WebLogic Collaborate to implement the changes.

2. Stop and restart the WebLogic Server.

   Note: All the WebLogic Collaborate components, such as the JMS queue, the repository, persistent storage, and administration, must use the same database and same JDBC connection pool.

   When you ran the Verification example, as described in the *BEA WebLogic Collaborate Installation Guide*, the example software set up the database. If you are using the same database, then you do not need to perform this step. However, to use a different database, you need to set it up.

   a. Install the JDBC driver.

      The WebLogic Server installation includes the Oracle and Cloudscape JDBC drivers. Therefore, you do not need to install a JDBC driver if you are using either the Oracle or Cloudscape database. If you are using Microsoft SQL Server, download and install the WebLogic jDriver for SQL Server as described in *Installing and Using WebLogic jDriver for MicroSoft SQL Server*.

   b. Create the database.

   Warning: When you create the database, you delete any already-existing database.

      Run the `createDB.cmd` (Windows) or `createDB.sh` (UNIX) file, which is located in the `bin` subdirectory of your WebLogic Collaborate installation directory. Use the following command syntax to specify your database:

      ```
      createDB cloudscape|oracle|mssql
      ```

   c. Run the Bulk Loader utility to load data into the repository.

   Warning: When you use the Bulk Loader to load data into the repository, you delete any already-existing data.

      For information about the Bulk Loader, see Chapter 3, “Working with the Bulk Loader.”

Configuring the Java Message Service Queue

Java Message Service (JMS) enables Java programs to exchange messages with other Java programs. WebLogic Collaborate uses the WebLogic Server implementation of the JMS queue. For more information about JMS, see "Managing JMS" in the *BEA WebLogic Server Administration Guide*. Even though the section describes how to create a database for JMS, you need only to create database tables.

Note: You do not need to perform most of the configuration tasks described in the "Configuring WebLogic JMS" section in the WebLogic Server documentation:

- Configure a JMS Server for dynamic queue registration. The name of the JMS Server should be `WLCJMSServer`. To see a sample of the configuration file after configuring `WLCJMSServer`, see the hub's config file in $WLC_HOME/hub/config/mydomain
- If you are using persistence and recovery, you need to set the data source for the JMS queue as described in "Procedure for Configuring Persistence and Recovery" on page 5-6 in Chapter 5, "Configuring Persistence and Recovery."
- You do not need to create a JDBC connection pool. You already did this when you configured the repository as described in "Configuring the Repository" on page 2-5.

You do not need to define connection factories, topics, queues, or durable subscribers. WebLogic Collaborate creates these JMS components.

Configuring and Running the C-Hub Administration Console

To configure and run the C-Hub Administration Console:

1. Use the WebLogic Server Administration Console to define the C-Hub Administration Console Web Application.

   The C-Hub Administration Console is a J2EE Web application. The file for the C-Hub Administration Console is `hubadmin.war`, which is located in the `lib` subdirectory of your WebLogic Collaborate installation directory. For more information, see "Deploying and Configuring Web Applications" in the *BEA WebLogic Server Administration Guide*.

   The following shows the content of a completed configuration for a sample Console Web Application.

Listing 2-4 Code Sample for Console Web Application

```
<Application
  Name="WLCHubAdmin"
  Path="<WLC_HOME>/lib">
  <WebAppComponent
    Name="WLCHubAdmin"
    ServletReloadCheckSecs="1"
    Targets="myserver"
    URI="hubadmin.war"
    WebServers="myserver"
  />
</Application>
```

2. The C-Hub Administration Console requires a login to ensure that only authorized users can configure and monitor c-hubs. Specify the list of authorized users and their associated permissions in the `WLCAdmin` access control list (ACL).

   The following example shows how to specify the `WLCAdmin` ACL:

```
acl.hubconfig.WLCAdmin=admin
acl.hubmonitor.WLCAdmin=user2,user3
```

For more information about defining access control lists, see "Defining ACLs" in "Managing Security" in the *BEA WebLogic Server Administration Guide*.

3. Set JDBC connection pool permissions.

   In addition to the `WLCAdmin` ACL permissions, users who need to configure the c-hub must be allowed to reserve connections from the JDBC pool for the repository. In the previous example, the user `admin` must have this type of permission. For information about JDBC connection pool permissions, see "Configuring the Repository" on page 2-5.

4. Start the C-Hub Administration Console.

   Open a Web browser and go to the following URL:

   `http://host:port/WLCHubAdmin`

   In this URL, *`host:port`* specifies the location of the WebLogic Server that is hosting the c-hub.

   For information about using the C-Hub Administration Console, see "Using the C-Hub Administration Console," which is Part II in this document.

Sample C-Hub Sections in the config.xml File

The following listing shows c-hub sections from an example of the `config.xml` file.

Listing 2-5 Sample C-Hub Configuration Sections in the config.xml File

```
<StartupClass
    ClassName="com.bea.b2b.hub.Startup"
    Name="WLCStartup"
    Targets="myserver"
 />
 <JDBCTxDataSource
    JNDIName="WLCHub.DS"
    Name="WLCHub.DS"
    PoolName="wlcPool"
```

```
      EnableTwoPhaseCommit="true"
      Targets=”myserver”
 />
   <Application
     Name=”WLCHubAdmin”
     Path=”d:/bea/WLC/lib”
   >
     <WebAppComponent
       Name=”WLCHubAdmin”
       ServletReloadCheckSecs=”1”
       Targets=”myserver”
       URI=”hubadmin.war”
     />
   </Application>
<JDBCConnectionPool
      CapacityIncrement=”1”
      DriverName=”oracle.jdbc.driver.OracleDriver”
      InitialCapacity=”1”
      LoginDelaySeconds=”1”
      MaxCapacity=”10”
      Name=”wlcPool”
      Properties=”user=scott;password=tiger”
      RefreshMinutes=”0”
      ShrinkPeriodMinutes=”15”
      ShrinkingEnabled=”true”
      Targets=”myserver”
      URL=”jdbc:oracle:thin:@fushigi.beasys.com:1521:fush816”
   />
   <JMSServer
     Name=”WLCJMSServer”
     Targets=”myserver”
   />
```

3 Working with the Bulk Loader

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

- Terminology
- Importing Data into the Repository
- Exporting Repository Data to a Repository Data File
- Deleting Repository Data
- Working with the Bulk Loader Configuration File
- Working with the Repository Data File
- Checking Data

Warning: When you use the Bulk Loader to modify the repository, these changes do not take effect until you reboot your WebLogic Collaborate system or you use the C-Hub Administration Console to make a change in a related area. Therefore, it is recommended that you do not use the Bulk Loader while the c-hub is up and running.

In addition to using the Bulk Loader, you can also use the C-Hub Administration Console to transfer data to and from the repository as described in "Using the C-Hub Administration Console," which is Part II of this document.

Terminology

The term “data element” refers to an element in the repository and the term “XML element” refers to a term in an XML file. The term “attribute” refers to an attribute for an XML element. The term “value” can refer to the value for an attribute or the value for a data element. For example:

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

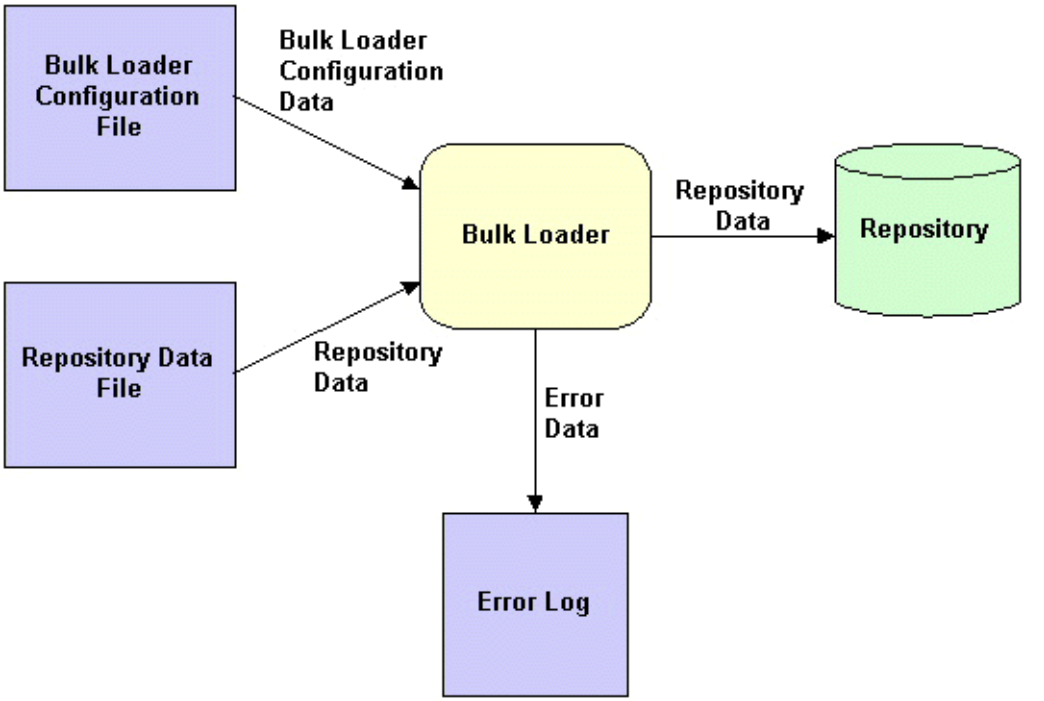
In these lines of code from a repository data file:

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

Importing Data into the Repository

As the following figure shows, the Bulk Loader reads the Bulk Loader configuration file to transfer data from a repository data file, which is in XML, into the repository. If the Bulk Loader detects any errors during this procedure, it creates an error log.

Figure 3-1 Using the Bulk Loader to Import Repository Data



The following sections describe how to import repository data:

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

How the Bulk Loader Imports Data

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

1. If the c-hub specified in the repository data file does not exist in the repository, the Bulk Loader creates the c-hub.
2. If the c-hub specified in the repository data file already exists in the repository, the Bulk Loader uses the following logic to process each data element that is represented 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 has one or more different data element values, attributes, or attribute values from the corresponding XML element in the repository data file, the Bulk Loader re-creates the data element using the logic described in the following table for each data element value and each attribute.

Table 3-1 Logic for Processing an Attribute

Value or Attribute Exists in 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 Table 3-2.
No	`REQUIRED`	The Bulk Loader considers the data to be invalid. For more information, see “Checking Data” on page 3-19.

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

Table 3-2 Default Values for Special IMPLIED Attributes

Data Element	Attribute	Default Value
c-hub	`large-msg-support-on`	`OFF`
c-hub	`show-hidden`	`OFF`
c-space	`status`	`ENABLED`
trading partner	`status`	`ENABLED`

Procedure for Importing Data into the Repository

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

Warning: When you use the Bulk Loader to modify the repository, these changes do not take effect until you reboot your WebLogic Collaborate system or you use the C-Hub Administration Console to make a change in a related area. Therefore, it is recommended that you do not use the Bulk Loader while the c-hub is up and running.

1. Create a Bulk Loader configuration file.

   In the Bulk Loader configuration file, include the `load-processing-parameters` XML element that instructs the Bulk Loader to import data from the repository data file into the repository. For information about creating the Bulk Loader configuration file, see "Working with the Bulk Loader Configuration File" on page 3-14.

2. Create a repository data file.

   For information about creating the repository data file, see "Working with the Repository Data File" on page 3-18.

3. To import the data from the repository data file into the repository, enter the following command:

   UNIX: `bulkloader.sh` *`cfg_file`*

   Windows: `bulkloader` *`cfg_file`*

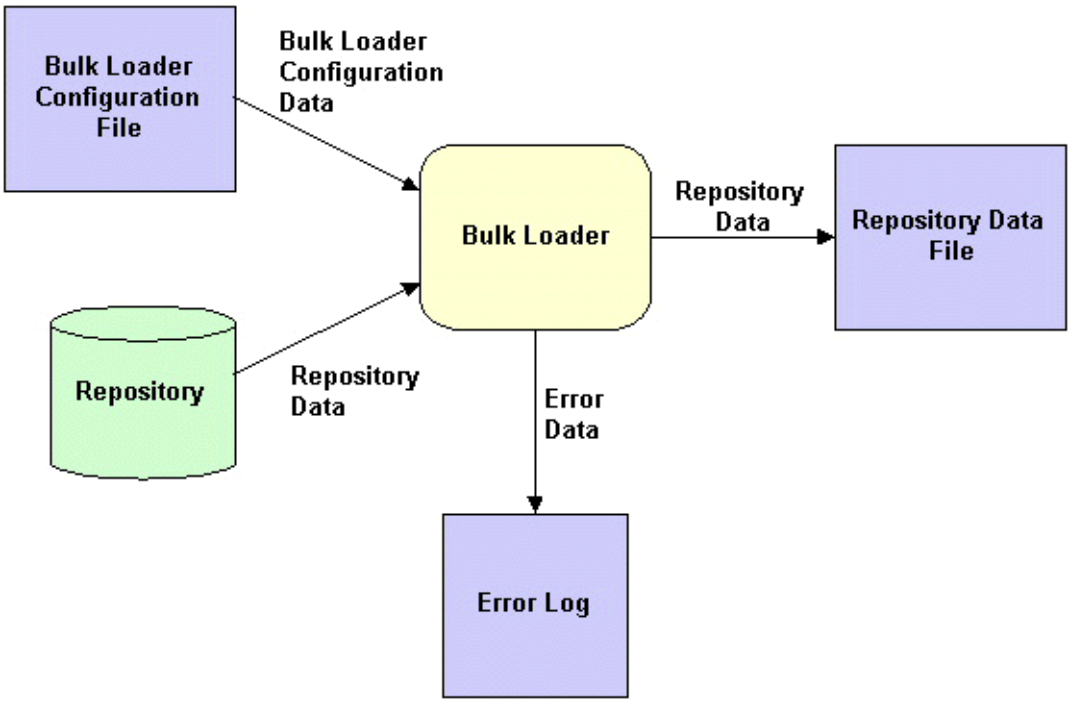
   In this command syntax, *`cfg_file`* is the pathname of the Bulk Loader configuration file that you created in Step 1. The Bulk Loader configuration file specifies the pathname of the repository data file that you created in Step 2. The Bulk Loader uses the same command for importing, exporting, and deleting. The Bulk Loader configuration file indicates which action to take.

   While importing data, the Bulk Loader checks for errors as described in "Checking Data" on page 3-19.

Exporting Repository Data to a Repository Data File

As the following figure shows, the Bulk Loader reads the Bulk Loader configuration file to transfer data from the repository to an XML repository data file. If the Bulk Loader detects any errors during this procedure, it creates an error log.

Figure 3-2 Using the Bulk Loader to Export Repository Data



The following sections describe how to export repository data:

- Full and Partial Repository Exports
- Short and Long Repository Exports
- Procedure for Exporting Repository Data

Full and Partial Repository Exports

When you export data from the repository, you can export all the data or you can export a portion of the data:

- A **full** repository export means that you are exporting all the data.
- A **partial** repository export means that you are exporting 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 export. The Bulk Loader uses the `entities` values to traverse the repository to the specified data elements.

For an example of a Bulk Loader configuration file that specifies a partial repository export, see Listing 3-5. For a complete description of the `entities` XML element, see `BulkLoader.dtd`, which is in the `dtd` subdirectory of your WebLogic Collaborate installation directory.

The following sections describe how to identify the data elements to export during a partial export:

- Exporting a Specific Instance of a Data Element
- Exporting All Instances of a Data Element

Exporting a Specific Instance of a Data Element

You can identify a specific data element instance in the Bulk Loader configuration file. For example, to export a specific c-space, identify the name of the c-hub and the name of the c-space as shown in the following listing. This strategy applies to all types of data elements, including c-hubs.

Listing 3-1 Bulk Loader Configuration File for Exporting a Specific C-Space

```
<?xml version="1.0"?>
<!DOCTYPE bulkloader SYSTEM "BulkLoader.dtd">
<bulkloader>
    <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>
            <c-hub name="MainHub">
                <c-space name="PriceSpace">
                </c-space>
            </c-hub>
        </entities>
    </unload-processing-parameters>
</bulkloader>
```

Exporting All Instances of a Data Element

If you do not identify a specific data element in the Bulk Loader configuration file, the Bulk Loader exports all such data elements. For example, to export all c-spaces in a certain c-hub, identify the c-hub and do not identify the name of the c-space as shown in the following listing. This strategy applies to all types of data elements, including c-hubs.

Listing 3-2 Bulk Loader Configuration File for Exporting All C-Spaces in a C-Hub

```
<?xml version="1.0"?>
<!DOCTYPE bulkloader SYSTEM "BulkLoader.dtd">
<bulkloader>
    <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>
            <c-hub name="MainHub">
                <c-space/>
            </c-hub>
        </entities>
    </unload-processing-parameters>
</bulkloader>
```

Short and Long Repository Exports

When you export data from the repository, you can export to the short format or the long format:

- The **short** (standard) format organizes the output data to represent the user's view of the repository.
- The **long** (extensive) format organizes the output data as a snapshot of the repository and is useful for migrating repository data from one database to another.

By default, the Bulk Loader exports repository data in the short format. To specify the long format, set the `format` attribute in the `unload-processing-parameters` in the Bulk Loader configuration file to "`long`."

Because the long format includes internal repository data in addition to the values for the 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 3-4. For a complete description of the `format` attribute, see `BulkLoader.dtd`, which is in the `dtd` subdirectory of your WebLogic Collaborate installation directory.

Procedure for Exporting Repository Data

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

1. Create a Bulk Loader configuration file.

   In the Bulk Loader configuration file, include the `unload-processing-parameters` XML element that instructs the Bulk Loader to export data from the repository to a repository data file. For information about creating the Bulk Loader configuration file, see "Working with the Bulk Loader Configuration File" on page 3-14.

2. To export data from the repository to a repository data file, enter the following command:

   UNIX: `bulkloader.sh` *`cfg_file`*

   Windows: `bulkloader` *`cfg_file`*

   In this command syntax, *`cfg_file`* is the pathname of the Bulk Loader configuration file that you created in Step 1. The Bulk Loader configuration file specifies the pathname of the repository data file into which the Bulk Loader exports the data. The Bulk Loader uses the same command for importing,

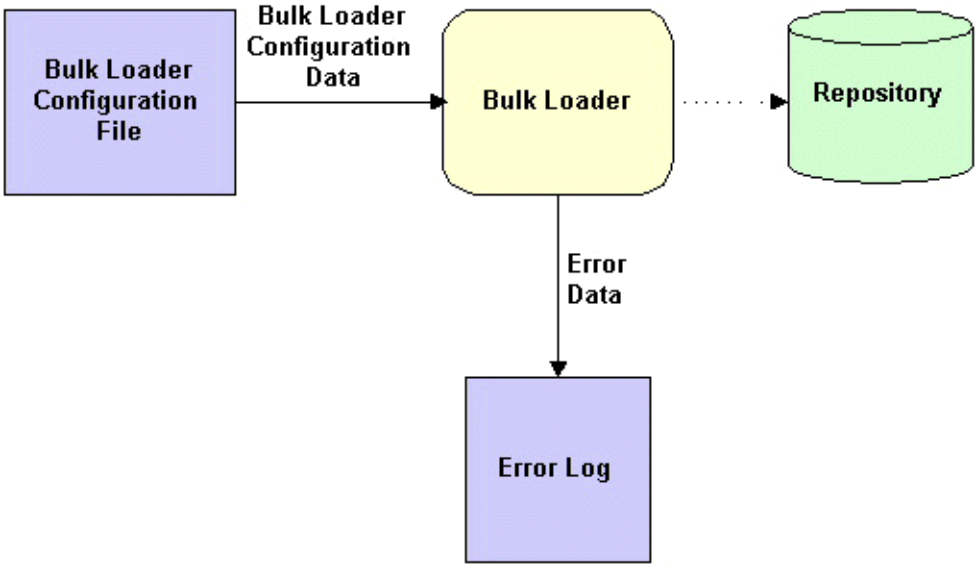
exporting, and deleting. The Bulk Loader configuration file indicates which action to take.

While exporting data, the Bulk Loader checks for errors as described in "Checking Data" on page 3-19.

Deleting Repository Data

As the following figure shows, the Bulk Loader reads the Bulk Loader configuration file to delete data from the repository. The dotted line in the figure indicates that the Bulk Loader affects the repository, but does not send any data to it. If the Bulk Loader detects any errors during this procedure, it creates an error log.

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



To delete repository data:

Warning: When you use the Bulk Loader to modify the repository, these changes do not take effect until you reboot your WebLogic Collaborate system or you use the C-Hub Administration Console to make a change in a related area. Therefore, it is recommended that you do not use the Bulk Loader while the c-hub is up and running.

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 Repository Data to a Repository Data File" on page 3-7.

2. Create a Bulk Loader configuration file.

   In the Bulk Loader configuration file, include the `delete-processing-parameters` XML element that instructs the Bulk Loader to delete data from the repository. For information about creating the Bulk Loader configuration file, see "Working with the Bulk Loader Configuration File" on page 3-14.

3. To delete the data from the repository, enter the following command:

   UNIX: `bulkloader.sh` *`cfg_file`*

   Windows: `bulkloader` *`cfg_file`*

   In this command syntax, *`cfg_file`* is the pathname of the Bulk Loader configuration file that you created in Step 2. The Bulk Loader uses the same command for importing, exporting, and deleting. The Bulk Loader configuration file indicates which action to take.

   While deleting data, the Bulk Loader checks for errors as described in "Checking Data" on page 3-19.

Working with the Bulk Loader Configuration File

The Bulk Loader configuration file is an XML file that uses the `BulkLoader.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 `BulkLoader.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 Deleting Data

Bulk Loader Configuration File for Importing Data

The following listing is an example Bulk Loader configuration file for importing data into the repository. In addition to specifying values that are required in any Bulk Loader configuration file (DTD, database URL, database driver, database user ID, database password), this example also specifies:

- `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 or not to delete all data from the repository before performing the repository import.

- `transaction-level`—The value for this attribute specifies what actions to take if the Bulk Loader detects an error. For more information, see "Checking Data Integrity" on page 3-21.
- `xml-file-name`—The value for this element specifies the pathname of the repository data file from which to import the data.

Listing 3-3 Bulk Loader Configuration File for Importing Data into the Repository

```
<?xml version="1.0"?>
<!DOCTYPE bulkloader SYSTEM "BulkLoader.dtd">
<bulkloader>
    <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>
</bulkloader>
```

Bulk Loader Configuration File for Exporting Data

The following listing is an example Bulk Loader configuration file for exporting data from the repository. In addition to specifying values that are required in any Bulk Loader configuration file (DTD, database URL, database driver, database user ID, database password), this example also specifies:

- `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 to export data in the long (extensive) format or in the short (standard) format. For more information, see "Short and Long Repository Exports" on page 3-10.
- `xml-file-name`—The value for this element specifies the pathname of the repository data file to which the data is exported.

Listing 3-4 Bulk Loader Configuration File for Performing a Full Export from the Repository

```
<?xml version="1.0"?>
<!DOCTYPE bulkloader SYSTEM "BulkLoader.dtd">
<bulkloader>
    <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>
</bulkloader>
```

The following listing is an example Bulk Loader configuration file for exporting a subscription from the repository. In addition to specifying values that are required in any Bulk Loader configuration file (DTD, database URL, database driver, database user ID, database password), this example also specifies:

- `unload-processing-parameters`—This XML element tells the Bulk Loader to export data from the repository to the repository data file.
- `xml-file-name`—The value for this element specifies the pathname of the repository data file into which to import the data.
- `entities`—This XML element specifies the data elements to export from the repository.

Listing 3-5 Bulk Loader Configuration File for Performing a Partial Export from the Repository

```
<?xml version="1.0"?>
<!DOCTYPE bulkloader SYSTEM "BulkLoader.dtd">
<bulkloader>
    <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>
```

```
            <c-hub name="MainHub">
                <c-space name="PriceSpace">
                    <collaborator trading-partner-name="TPFailureReportAdmin">
                        <subscription
                            conversation-name="Conversation1"
                            conversation-version="1.1"
                            role-name="FailureReportAdmin">
                        </subscription>
                    </collaborator>
                </c-space>
            </c-hub>
        </entities>
    </unload-processing-parameters>
</bulkloader>
```

Bulk Loader Configuration File for Deleting Data

The following listing is an example Bulk Loader configuration file for deleting a subscription from the repository. In addition to specifying values that are required in any Bulk Loader configuration file (DTD, database URL, database driver, database user ID, database password), this example also specifies:

- `delete-processing-parameters`—This XML element tells the Bulk Loader to delete data from the repository.
- `transaction-level`—The value for this attribute specifies what actions to take if the Bulk Loader detects an error. For more information, see "Checking Data Integrity" on page 3-21.
- `entities`—This XML element specifies the data elements to delete from the repository.

Listing 3-6 Bulk Loader Configuration File for Deleting Data from the Repository

```
<?xml version="1.0"?>
<!DOCTYPE bulkloader SYSTEM "BulkLoader.dtd">
<bulkloader>
    <delete-processing-parameters 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>
        <entities>
            <c-hub name="MainHub">
                <c-space name="PriceSpace">
                    <collaborator trading-partner-name="Admin">
                        <subscription
                            conversation-name="Conversation1"
                            conversation-version="1.1"
                        </subscription>
                    </collaborator>
                </c-space>
            </c-hub>
        </entities>
    </delete-processing-parameters>
</bulkloader>
```

Working with the Repository Data File

The repository data file is an XML file that uses the `WLCHub.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 `WLCHub.dtd` as the DTD file. The following listing shows a repository data file that creates a new conversation and subscribes trading partners to the conversation.

Listing 3-7 Repository Data File for a New Conversation

```
<?xml version="1.0"?>
<!DOCTYPE c-hub SYSTEM "WLCHub.dtd">
<c-hub name="supplies-hub">

    <c-space name="supplies-space">
        <collaborator trading-partner-name="Partner1">
            <subscription
                conversation-name="Conversation1"
                conversation-version="1.0"
                role-name="buyer">
            </subscription>
        </collaborator>
```

```
        <collaborator trading-partner-name="Partner2">
            <subscription
                conversation-name="Conversation1"
                conversation-version="1.0"
                role-name="seller">
            </subscription>
        </collaborator>
    </c-space>

    <conversation-def
        name="Conversation1"
        version="1.0"
        default-durability="NONPERSISTENCE"
        default-conversation-timeout="0">
        <role name="buyer">
            <in-message message-def-name="reply.dtd"/>
            <out-message message-def-name="request.dtd"/>
        </role>
        <role name="seller">
            <in-message message-def-name="request.dtd"/>
            <out-message message-def-name="reply.dtd"/>
        </role>
    </conversation-def>

</c-hub>
```

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 file to `wlc.log.`*`yyyy.mm.dd.hh.mi.ss`*. The following table describes this filename format.

Table 3-3 Error Log Filename Format

Field	Description
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 processing 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 3-4 Files That the Bulk Loader Validates

Task	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 During an Import or a Delete

For an import or a delete, you can set the `transaction-level` attribute in the Bulk Loader configuration file as shown in Listing 3-3. If you do not set the `transaction-level`, the Bulk Loader uses the `default` value. The Bulk Loader takes the following actions depending on the value of `transaction-level`:

- **`all`**—The Bulk Loader performs a single transaction for all of 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 as it was before the Bulk Loader started importing or deleting data.
- **`default`**—The Bulk Loader performs a separate transaction for each of the following types of high-level data elements:
  - C-space
  - Collaborator
  - Conversation definition
  - Trading partner
  - Document definition
  - Transport protocol
  - Logic plug-in

- Business protocol definition
- Extended property set
- Message definition

If the Bulk Loader detects invalid data during a transaction, it rolls back the current transaction and then performs the next transaction (for the next high-level data element).

If the Bulk Loader detects invalid data for a c-hub data element at the c-hub level (such as a c-hub attribute), it rolls back all transactions that it performed for that c-hub 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

4 Configuring Security

This following sections describe how to configure security in BEA WebLogic Collaborate:

- Introduction to the WebLogic Collaborate Security Model
- Security Terminology
- Configuring the SSL Protocol and Mutual Authentication
- Defining Users and Groups
- Defining Access Control Lists
- Specifying Information in the Repository
- Working with the WLCCertAuthenticator Class
- Configuring WebLogic Collaborate to Use an HTTP Proxy Server

For information about configuring administration security, see “Configuring and Running the C-Hub Administration Console” on page 2-11.

Introduction to the WebLogic Collaborate Security Model

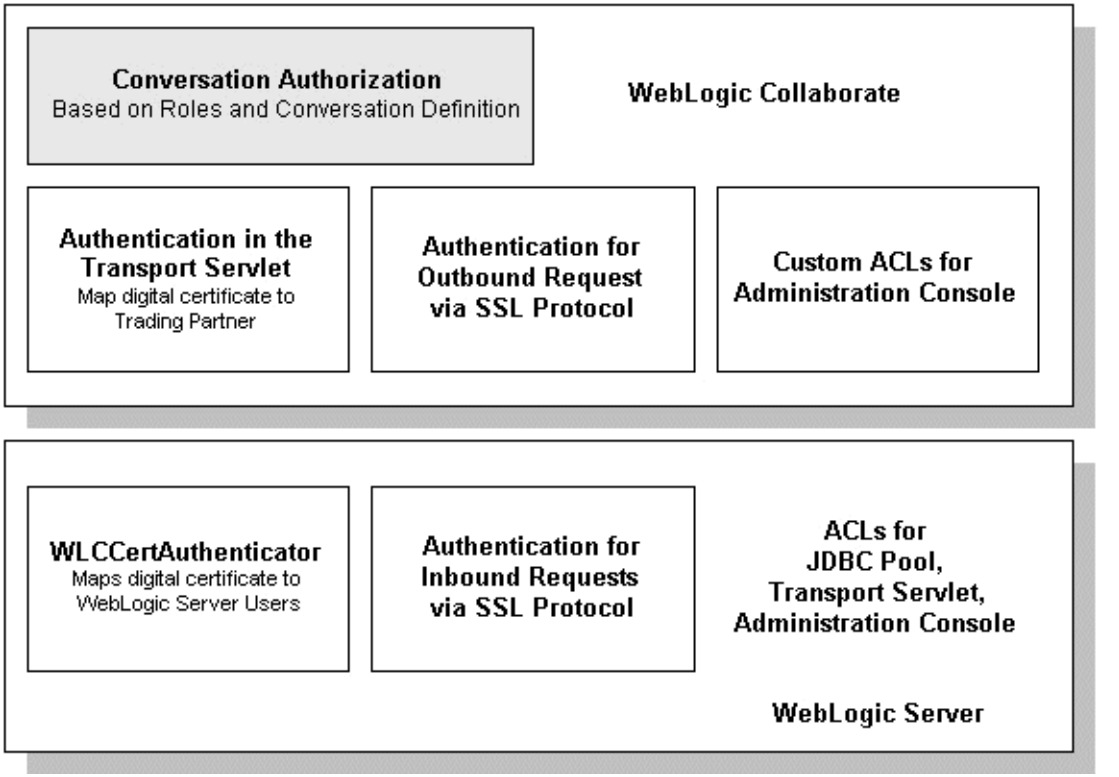
The WebLogic Collaborate security model uses the security features of the underlying BEA WebLogic Server™ platform.

Authentication is the process of verifying a principal's identity before completing a connection. Authentication protects who gets access to the WebLogic Collaborate environment. WebLogic Collaborate uses the following methods to perform authentication:

- Username and password—Human users (administrators) use a username and password to prove their identity.
- Digital certificates—Principals in the WebLogic Collaborate environment other than human users use digital certificates to prove their identity to other principals.
- Secure sockets layer (SSL)—The SSL protocol provides data integrity and confidentiality to the connections between principals.

In the WebLogic Collaborate environment, *authorization* protects who has access to the available resources. Permission to access WebLogic Collaborate resources is assigned through access control lists and roles. The following figure illustrates the WebLogic Collaborate security model.

Figure 4-1 WebLogic Collaborate Security Model



For information about the WebLogic Server security features used by WebLogic Collaborate, see "Configuring the SSL Protocol" and "Defining ACLs" in "Managing Security" in the *BEA WebLogic Server Administration Guide*.

Security Terminology

The following sections describes the security terminology used by the WebLogic Collaborate security model:

- Transport Servlet
- Resources
- Principals, Users, and Groups
- Authorization
- Digital Certificates
- Certificate Authority
- SSL Protocol
- Authentication

Transport Servlet

A transport servlet is the entry point into a WebLogic Collaborate system for communication between:

- The c-hub and c-enablers that use XOCP
- The c-hub and non-XOCP clients

A transport servlet is dynamically registered in the WebLogic Server environment for each c-space and c-enabler.

Resources

Resources are objects in the WebLogic Collaborate environment. Some of the WebLogic Collaborate resources are:

- Transport servlets on the c-hub and c-enabler
- C-Hub Administration Console
- C-Enabler Administration Console
- JDBC connection pool
- Conversations

Principals, Users, and Groups

Principals are objects that need access to the WebLogic Collaborate environment and resources. WebLogic Collaborate principals include:

- Trading partners
- Human users—c-hub administrators and c-enabler administrators
- C-hub servers

Principals are granted access to the WebLogic Collaborate environment and resources through authentication and authorization mechanisms. Principals in WebLogic Collaborate map to WebLogic Server users.

If the c-hub or c-enabler can prove the identity of the WebLogic Server user, the c-hub or c-enabler associates the user with a thread that executes code on behalf of the user. Before the thread begins executing code, the c-hub or c-enabler checks pertinent access control lists (ACLs) to make sure the WebLogic Server user has the proper permission to continue.

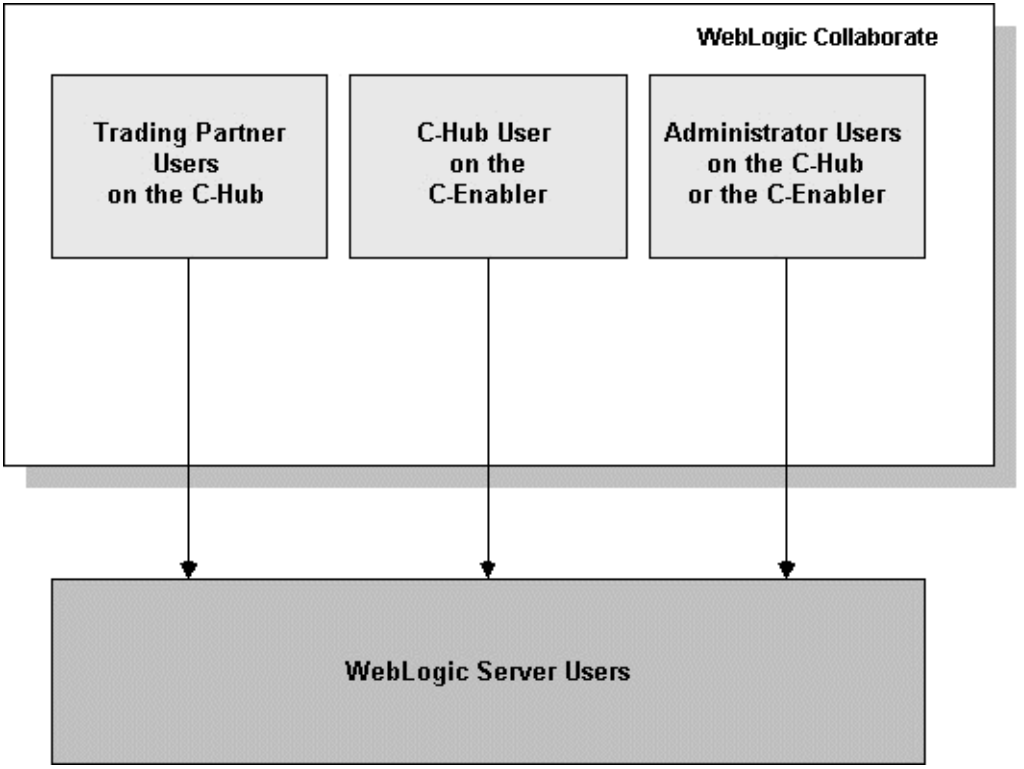
WebLogic Collaborate can have the following types of WebLogic Server users:

- Trading partner users on the c-hub
- A c-hub user on the c-enabler

- A c-hub administrator user
- A c-enabler administrator user

The following figure illustrates the relationship between principals in WebLogic Collaborate and WebLogic Server users.

Figure 4-2 WebLogic Collaborate Principals



Groups are sets of WebLogic Server users. Groups provide an efficient way to manage large numbers of users because an administrator can specify permissions for an entire group at one time.

Authorization

Authorization is the process of allowing a WebLogic Collaborate principal access to particular WebLogic Collaborate resources. The authorization model in WebLogic Collaborate is based on an ACL and permission mechanism and role-based authorization control.

The following sections describe how authorization is used in the WebLogic Collaborate environment:

- Access Control Lists
- Transport Servlet, JDBC Connection Pool, and Administration Consoles
- Conversations

Access Control Lists

ACLs are data structures with multiple entries that guard access to WebLogic Collaborate resources. An ACL grants permission on a resource, or class of resources, to a list of users and groups. An ACL includes a list of `AclEntries`, each with the set of permissions for a particular user or group.

Permissions represent privileges required for accessing a resource and are related to the resource they protect. The exact permissions available depend on the type of resource the ACL protects. For example, there are permissions to send and receive files, delete files, read and write files, and load servlets.

You define ACLs for the following WebLogic Collaborate resources:

- The transport servlet
- The C-Enabler Administration Console
- The C-Hub Administration Console
- The JDBC connection pool

Transport Servlet, JDBC Connection Pool, and Administration Consoles

Authorization allows a user to access WebLogic Collaborate resources based on ACLs for the resources. The WebLogic Collaborate resources that require authorization are:

- Transport servlets
- C-Hub Administration Console
- C-Enabler Administration Console
- JDBC connection pool

WebLogic Collaborate can have the following access control policies:

- Trading partner users need access to the transport servlet on the c-hub and the JDBC connection pool.
- The c-hub user on the c-enabler needs access to the transport servlet on the c-enabler for the purpose of sending messages to the c-enabler.
- The c-hub administrator users need access to the C-Hub Administration Console. This access is achieved by configuring custom ACLs for the C-Hub Administration Console.
- The c-enabler administrator users need access to the C-Enabler Administration Console. This access is achieved by configuring custom ACLs for the C-Enabler Administration Console.

Conversations

WebLogic Collaborate conversations use a role-based authorization whereby a trading partner is given access to a conversation based on a role that is defined in subscriptions for that trading partner.

A particular trading partner in a c-space can send and receive certain types of documents as defined by the trading partner's role in a specific conversation. Authorization to send a document in a conversation is based on the subscriptions of the trading partner. Subscription information identifies:

- Trading partner
- Name of the conversation

- Roles for all trading partners in the conversation
- Document types

The conversation definition identifies:

- Conversation name
- Conversation version
- Allowed roles
- Allowed document types

The c-hub compares the information in the subscription to the conversation definition. Based on this comparison, the c-hub allows a trading partner to participate in conversations by sending and receiving messages.

For example, suppose `trading partner x` has a subscription to a conversation named `aConv` with the role of `buyer`. The `aConv` conversation has two roles `buyer` and `seller`. The `aConv` conversation specifies that the `buyer` role has one incoming document (`reply.dtd`) and one outgoing document (`request.dtd`).

When `trading partner x` joins the c-space, it creates a conversation handler for the `aConv` conversation with a role of `buyer`. The c-hub prevents `trading partner x` from participating with a role of `seller`. In addition, the c-hub permits `trading partner x` to send only documents of type `request.dtd`.

To configure the authorization of a trading partner in a c-space, define the following information:

- Trading partner subscriptions, including the name and version of the conversation and the role of the trading partner in the conversation.
- Conversation definitions including the conversation name, the conversation version, conversation roles, and input and output document types for these roles.

You can use the C-Hub Administration Console to define this information as described in Chapter 15, “Setting Up Conversations,” and in “Step 4. Assign Subscriptions (Roles and Conversations) to a Trading Partner.” on page 16-12 in Chapter 16, “Working with C-Spaces.” You can also use the Bulk Loader to import this information into the repository as described in “Importing Data into the Repository” on page 3-3 in Chapter 3, “Working with the Bulk Loader.”

Digital Certificates

Digital certificates are electronic documents used to uniquely identify principals and objects over networks such as the Internet. A digital certificate securely binds the identity of a user or object, as verified by a trusted third party known as a certificate authority, to a particular public key. The combination of the public key and the private key provides a unique identity to the owner of the digital certificate.

Digital certificates allow verification of the claim that a specific public key does in fact belong to a specific user or entity. A recipient of a digital certificate can use the public key contained in the digital certificate to verify that a digital signature was created with the corresponding private key. If such verification is successful, this chain of reasoning provides assurance that the corresponding private key is held by the subject named in the digital certificate, and that the digital signature was created by that particular subject.

A digital certificate typically includes a variety of information, such as:

- The name of the subject (holder, owner) and other identification information required to uniquely identify the subject, such as a URL or an email address.
- The subject's public key.
- The name of the certificate authority that issued the digital certificate.
- A serial number.
- The validity period (or lifetime) of the digital certificate (defined by a start date and an end date).

The most widely accepted format for digital certificates is defined by the ITU-T X.509 international standard. Thus, digital certificates can be read or written by any application complying with the X.509 standard. The public key infrastructure (PKI) in WebLogic Server recognizes digital certificates that comply with X.509 version 3, or X.509v3.

Certificate Authority

Digital certificates are issued by a certificate authority. Any trusted third-party organization or company that is willing to vouch for the identities of those to whom it issues digital certificates and public keys can be a certificate authority. When a certificate authority creates a digital certificate, the certificate authority signs it with its private key, to ensure the detection of tampering. The certificate authority then returns the signed digital certificate to the requesting subject.

The subject can verify the signature of the issuing certificate authority by using the public key of the certificate authority. The certificate authority makes its public key available by providing a digital certificate issued from a higher-level certificate authority attesting to the validity of the public key of the lower-level certificate authority. This hierarchy of certificate authorities is terminated by a self-signed digital certificate known as the root certificate.

The recipient of an encrypted message can develop trust in the private key of a certificate authority recursively, if the recipient has a digital certificate containing the public key of the certificate authority signed by a superior certificate authority whom the recipient already trusts. In this sense, a digital certificate is a stepping stone in digital trust. Ultimately, it is necessary to trust only the public keys of a small number of top-level certificate authorities. Through a chain of digital certificates, trust in a large number of users' digital signatures can be established.

Thus, digital signatures establish the identities of communicating entities, but a digital signature can be trusted only to the extent that the public key for verifying the digital signature can be trusted.

SSL Protocol

The SSL protocol provides secure connections by enabling two applications connecting over a network connection to authenticate the other's identity and by encrypting the data exchanged between the applications. An SSL connection begins with a handshake during which the applications exchange digital certificates, agree on the encryption algorithms to use, and generate encryption keys used for the remainder of the session.

The SSL protocol provides the following security features:

- Server authentication—The server uses its digital certificate, issued by a trusted certificate authority, to authenticate itself to clients.
- Client authentication—Optionally, clients might be required to authenticate themselves to the server by providing their own digital certificates. This type of authentication is also referred to as mutual authentication. The authentication model in WebLogic Collaborate uses mutual authentication.
- Data privacy—All client requests and server responses are encrypted to maintain the confidentiality of the data exchanged over the network.
- Data integrity—Data that flows between a client and server is protected from a third party's tampering.

The SSL protocol is used to implement link-level encryption of messages sent between the c-hub and the c-enabler.

Administrators use a Web browser to access the C-Hub and C-Enabler Administration Consoles. You can use the Hypertext Transfer Protocol with SSL (HTTPS) to secure this type of network communication.

Authentication

Authentication is the process by which WebLogic Collaborate establishes the identity of a principal. Digital certificates with mutual authentication over SSL or HTTPS are used between the c-enabler (on behalf of the trading partner) and the c-hub. Both the c-hub and the c-enabler examine and validate the digital certificates against defined security information. Human users use a username/password combination to authenticate themselves to the WebLogic Collaborate environment.

Configuring the SSL Protocol and Mutual Authentication

To configure the c-hub to use the SSL protocol and mutual authentication, perform the following steps:

1. Obtain a digital certificate for the c-hub. WebLogic Collaborate ships four digital certificates and private keys (one for the c-hub, one for the c-enabler, and two for trading partners) in the *WLC_HOME*`/examples/security/certificates` directory. The directory also contains a digital certificate for the root certificate authority.

   Note: The digital certificates and private keys shipped with WebLogic Collaborate are for demonstration purposes only. Before using WebLogic Collaborate in a deployed, production environment, obtain digital certificates and private keys from a security vendor or an in-house certificate authority.

   For more information on Secure Socket Layers (SSL), see "Managing Security" in the *BEA WebLogic Server Administration Guide*.

2. In the WebLogic Administration Console, specify the following security settings:

   - SSL Port
   - Client root CA
   - Hub Certificate file
   - Hub Private Key file
   - Certificate for root CA

   The following listing is an example of the modified `config.xml` file, showing the changed SSL protocol and related authentication settings.

Listing 4-1 Example from config.xml of the SSL Protocol and Mutual Authentication for the C-hub

```
<SSL CertAuthenticator="com.bea.b2b.security.WLCCertAuthenticator"
        CertificateCacheSize="5"
```

```
 ClientCertificateEnforced=”true”
        Enabled=”true”
        HandlerEnabled=”true”
        ListenPort="SSL Port"
        Name=”myserver”

ServerCertificateFileName=Hub Certificate file
        ServerCertificateChainFileName=rest of the
digital certificates for Hub
        ServerKeyFileName=Hub private Key
        TrustedCAFileName=Certificate for root CA
      />
```

In the preceding listing:

- *SSL port* specifies the dedicated port on which the c-hub listens for SSL connections. The `config.xml` file for the c-hub has the SSL port set to 7002.
- *Client Root CA* specifies the name of the digital certificate for the certificate authority used to issue digital certificates for trading partners. Trading partners are required to present digital certificates issued by this certificate authority. The `config.xml` file for the c-hub has the client root CA set to `CA_cert.pem`.
- *SSL port* specifies the dedicated port on which the c-enabler listens for SSL connections. The config.xml file for the c-enabler has the SSL port set to 7502.
- *Hub Certificate file* specifies the name of the digital certificate for the c-hub. The `config.xml` file for the c-hub has the hub certificate file set to `hub_cert.pem`.
- *Hub Private Key* specifies the name of the private key for the c-hub. The `config.xml` file for the c-hub has the hub private key file set to `hub_key.pem`.
- *Certificate for root CA* specifies the name of the digital certificate for the certificate authority that issued the digital certificate for the c-hub.The `config.xml` file for the c-hub has the certificate for the root CA set to `CA_cert.pem`.

Defining Users and Groups

To control access to your web applications, you first need to define identities for users on the c-hub. Define the following types of users:

- Users that correspond to trading partners. These users must also be specified in the repository for each trading partner.
- A user for the c-hub administrator.

For complete information about defining users and groups, see "Defining Users" and "Defining Groups" in "Managing Security" in the *BEA WebLogic Server Administration Guide*.

Defining Access Control Lists

The Access Control List (ACL) for a resource determines whether a user or group can access a resource in WebLogic Collaborate. To define ACLs, you first create an ACL for a resource, next specify the permission for the resource, and then grant the permission to a specified set of users and groups. Each WebLogic Collaborate resource has one or more permissions for the resource that can be granted.

On the c-hub you need to define the following ACLs:

- On the transport servlet for trading partner users; the transport servlet has only an `execute` permission.
- On the JDBC connection pool for trading partner users; the JDBC connection pool has `reserve`, `reset`, and `shrink` permissions.
- On the C-Hub Administration Console for the c-hub administrator user; the C-Hub Administration Console has `hubconfig` and `hubmonitor` permissions.

For complete information about defining ACLs, see "Defining ACLs" in the "Managing Security" in the *BEA WebLogic Server Administration Guide*.

The following listing includes properties that define the c-hub ACLs. Note that the code example assumes the name of the transport servlet on the c-hub is `cspace1`.

Listing 4-2 C-Hub ACLs

```
#ACL for Transport Servlet
acl.execute.weblogic.servlet.Hub/SecurityCSpace=eng
#ACL for JDBC Connection Pool
acl.reserve.weblogic.jdbc.connectionPool.wlcPool=everyone
acl.reset.weblogic.jdbc.connectionPool.wlcPool=everyone
acl.shrink.weblogic.jdbc.connectionPool.wlcPool=everyone

#ACL for Administration Console
acl.hubconfig.WLCAdmin=admin
acl.hubmonitor.WLCAdmin=tp1
```

You can substitute groups for users in the ACLs. For more information on access control lists, see Defining ACLs in "Managing Security" in the *BEA WebLogic Server Administration Guide*.

Specifying Information in the Repository

The c-hub repository contains security information about the c-hub and the trading partners that access the c-hub. Repository information can be either configured through the C-Hub Administration Console or specified in a repository data file and then imported into the repository using the Bulk Loader.

Define the following security information for the c-hub:

- Certificate field name
- Server certificate field name
- Certificate location
- Private key location

Define the following security information for each trading partner that accesses the c-hub:

- Certificate field value
- WebLogic Server user name
- List of server certificate field values

For information about the C-Hub Administration Console, see "Creating a Trading Partner" on page 14-2 and "Defining Server-Side Security Values for a Trading Partner" on page 14-8 in Chapter 14, "Working with Trading Partners." For information about the Bulk Loader, see "Importing Data into the Repository" on page 3-3 in Chapter 3, "Working with the Bulk Loader."

The following listing contains an example of the security information you need to define for the c-hub and trading partners in the repository. For more information, see the `readme.htm` file in the *`WLC_HOME`*`/examples/security` directory.

Listing 4-3 Security Information for the Repository.

```
<c-hub
       ...
       certificate-location="<WLC_HOME>\examples\security
           \certificates\hub_cert.pem"
       private-key-location="<WLC_HOME>\examples\security
           \certificates\hub_key.pem"
       certificate-field-name="email"
       server-certificate-field-name="email"
       ...
/>

<trading-partner
       name="SecurityPartner1"
       email="partner1@bea.com"
       user-name="securityPartner1"
       certificate-field-value="e63914a52929a16dce3f6a5cb4bf67a2"
       <server-certificate
           certificate-field-value="enabler@bea.com"/>
</trading-partner>

<trading-partner
       name="SecurityPartner2"
       email="partner2@bea.com"
       user-name="securityPartner2"
```

```
        certificate-field-value="fb8b5a3a39d71c49817fc55384798707"
            <server-certificate
        certificate-field-value="enabler@bea.com"/>
</trading-partner>
```

Working with the WLCCertAuthenticator Class

The following sections describe how to work with the `WLCCertAuthenticator` class:

- Specifying the WLCCertAuthenticator Class
- Customizing the WLCCertAuthenticator Class

Specifying the WLCCertAuthenticator Class

WebLogic Collaborate contains an implementation of the `weblogic.security.acl.CertAuthenticator` class. On the c-hub, `CertAuthenticator` maps digital certificates from trading partners to their corresponding WebLogic Server users. On the c-enabler, `CertAuthenticator` maps the digital certificates from the c-hub to a corresponding WebLogic Server user.

Use the WebLogic Server Administration Console to configure the WebLogic Collaborate implementation of the certificate authority (`com.bea.b2b.security.WLCCertAuthenticator`) of the `weblogic.security.acl.CertAuthenticator` class. The following listing shows the property that defines the WebLogic Collaborate implementation of the `weblogic.security.acl.CertAuthenticator` class.

Listing 4-4 WebLogic WLCCertAuthenticator Class

```
<SSL CertAuthenticator="com.bea.b2b.security.WLCCertAuthenticator"
   CertificateCacheSize="5"
```

```
ClientCertificateEnforced="true"
       Enabled="true"
       HandlerEnabled="true"
       ListenPort="SSL Port"
       Name="myserver"
       ServerCertificateFileName=Hub Certificate file
       ServerCertificateChainFileName=rest of the digital certificates for Hub
       ServerKeyFileName=Hub private Key
       TrustedCAFileName=Certificate for root CA
     />
```

Customizing the WLCCertAuthenticator Class

The `WLCCertAuthenticator` is an implementation of the WebLogic Server `CertAuthenticator` class. The default implementation of the `WLCCertAuthenticator` class validates a trading partner and maps the digital certificate of the trading partner to the corresponding trading partner defined on the c-hub. You may want to extend this functionality to use mutual authentication for users other than trading partners. For example, you may want to modify the class to map a Web browser or Java client to a user on the c-hub.

The `WLCCertAuthenticator` class is called after an SSL connection has been established. The class can extract data from a digital certificate to determine the user which corresponds to the digital certificate.

For a code example of customizing the `WLCCertAuthenticator` class, see the Javadoc for the `WLCCertAuthenticator` class.

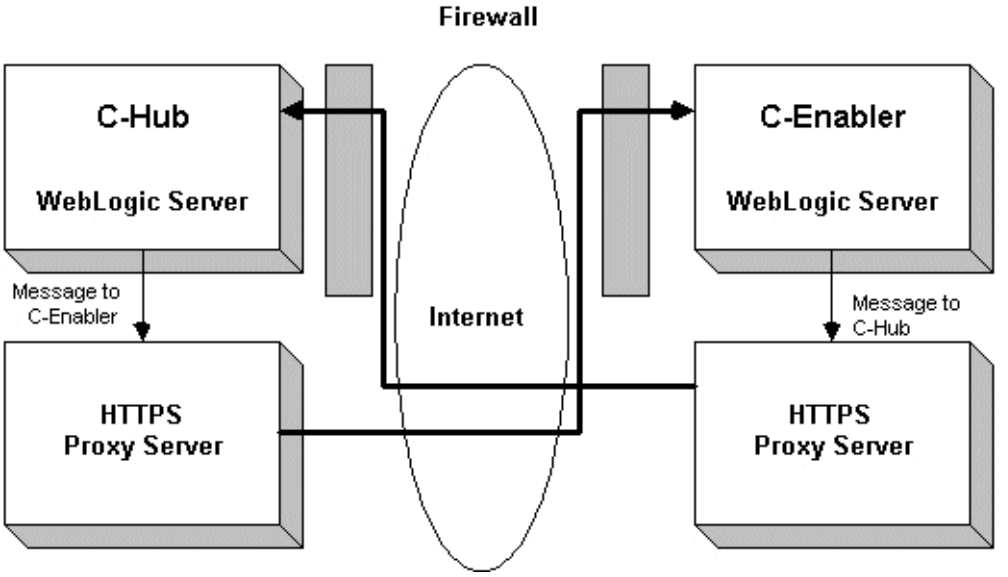
Configuring WebLogic Collaborate to Use an HTTP Proxy Server

If you are using WebLogic Collaborate in the security-sensitive environment, you may want to use WebLogic Collaborate behind a proxy server. A proxy server allows the c-hub and c-enabler to communicate across intranets or the Internet without compromising security. A proxy server is used to:

- Hide the local network addresses of the WebLogic Servers that host the c-hub or c-enabler from external hackers
- Restrict access to the external network
- Monitor external network access to the WebLogic Servers that host the c-hub or c-enabler

When proxy servers are configured on the local network WebLogic Collaborate uses, network traffic (SSL and HTTP) is tunneled through the proxy server to the external network. The following figure illustrates how a proxy server might be used in the WebLogic Collaborate environment.

Figure 4-3 Proxy Server



To configure a proxy server on the c-hub side of a network, define a host name and port number for the proxy server in the c-hub repository.

To configure a proxy server on the c-enabler side of a network, define a host and port for the proxy server in the c-enabler configuration file. For more information, see the *BEA WebLogic Collaborate C-Enabler Administration Guide*.

You need to add permissions to read and write the `ssl.proxyHost` and `ssl.proxyPort` system properties for the WebLogic Server. These system properties are stored in the `weblogic.policy` file which is located in the directory where you installed WebLogic Server. Add the following lines to the *`grant`* section of the `weblogic.policy` file.

```
permission java.util.PropertyPermission "ssl.proxyHost", "read, write";
permission java.util.PropertyPermission "ssl.proxyPort", "read, write";
```

CHAPTER

5 Configuring Persistence and Recovery

The following sections describe c-hub persistence and recovery:

- Persistence
- State Records
- Recovery
- Procedure for Configuring Persistence and Recovery

Persistence

If the recovery mode argument for the c-hub start-up class is set to `ON` for the c-hub, the c-hub saves the state of a c-hub component to persistent storage when it detects a state change for the component. The types of components for which the c-hub saves the component state are:

- C-hub
- C-space
- Trading partner

- Conversation
- Message

When the c-hub detects a state change for a message, the c-hub also saves the message or the message location. If large message support is enabled and the message is a large message, the c-hub saves the message location to persistent storage. Otherwise, the c-hub saves the message to persistent storage.

Persistent storage consists of a database that the c-hub accesses by means of a connection pool. Persistent storage and the repository share the same database. The c-hub creates a state record for each component that it saves to persistent storage. A state record is a row in a database table and represents an object state.

Multiple components can change state during the time that a message passes through the c-hub. The c-hub must update the persistent storage for these components as a group. If a message passes through the c-hub successfully, then the recorded changes are retained. If the message fails, then the c-hub discards the changes. To maintain and update component states as a group, the c-hub uses transactions.

The c-hub uses the database as persistent storage, which means that:

- The c-hub saves component states and message content or location from memory to the database only during normal run-time processing.
- The c-hub reads state records and message content or location from the database back into memory only during the restart/recovery phase of a new process instance in order to re-create a run-time image of the c-hub.

The c-hub persistence strategy is:

1. When the c-hub creates a component in memory, it also creates a state record for the component. When the c-hub creates a message component, it saves the message content or location in persistent storage in addition to the message state.
2. The c-hub maintains component state information in memory and synchronizes the in-memory state with the state record.
3. When a component state changes, the c-hub updates the corresponding state record. A message traveling through the system defines an action that causes the c-hub to create new components or update existing components. In turn, an action can spawn messages that define additional actions. And these actions cause more components to be created and updated. An update to a component state is complete only when the corresponding state record has been updated.

4. If the c-hub cannot successfully update a component state, the in-memory state becomes inconsistent with the corresponding state record. When this happens, the update task aborts and the c-hub instance crashes.

   Note: When a crash occurs, you must resolve the cause of the crash and reboot the WebLogic Collaborate process to recover the persistent data.

5. The only time a state record exists without a corresponding in-memory component state is during restart/recovery. In this case, the existence of the state record causes the c-hub to re-create the component. When the recovery process concludes, the c-hub updates the state record to indicate that it belongs to the new instance of the c-hub.

State Records

A state record is a row in a database table. The row contains a copy of the component state. You can use the C-Hub Administration Console to view the state of each component. For information about using the C-Hub Administration Console to

monitor components, see Chapter 18, "Monitoring the C-Hub." The following table describes the state records.

Table 5-1 State Records

State Record	Description
Hub State	State of the c-hub, such as active or inactive. The Hub State record contains the last known instance count. The instance count is a counter that increases each time the c-hub is successfully booted after a crash. The first time you boot the c-hub, the c-hub sets the instance count to 100. If the c-hub crashes and then is successfully rebooted, the c-hub increments the instance count. If the c-hub terminates in an orderly shutdown, all state records are deleted and the c-hub sets the instance count to 100 when you reboot it. The c-hub stores the instance number in each state record to indicate that the state record belongs to the currently active c-hub. During recovery, the c-hub uses the instance numbers to identify the components that it needs to recover. The c-hub updates the Hub State record: ■ After a successful c-hub boot or reboot ■ When the c-hub state changes ■ At c-hub shutdown
CSpace State	State of a c-space, such as active or inactive. The c-hub creates a CSpace State record when it creates a c-space. The c-hub updates the CSpace State record: ■ When a trading partner joins or leaves the c-space ■ After a successful recovery
Collaborator State	State of a trading partner, such as connected, registered, active, dropped out, or disconnected. The c-hub creates a Collaborator State record when a trading partner joins the c-space. The c-hub updates the Collaborator State record: ■ When the trading partner registers or unregisters for a conversation ■ When the trading partner leaves the c-space
Conversation State	State of an active conversation, such as initiated or terminated. The c-hub creates a Conversation State record when it creates a conversation. The c-hub updates the Conversation State record: ■ When a participant joins the conversation ■ When a participant leaves the conversation ■ When the conversation changes state The c-hub purges the Conversation State record after the conversation terminates.

Table 5-1 State Records

State Record	Description
Persistent Message Store	State of an outstanding message as it travels through the c-hub. The c-hub creates a Persistent Message Store record when it receives a message, and purges the Persistent Message Store record after it delivers the message to all recipients.

Recovery

When you restart a c-hub, it examines persistent storage for state records. The c-hub uses the presence or absence of state records to determine the last known state of the c-hub before termination. The type of termination was one of the following:

- *Orderly shutdown*—An orderly shutdown is the termination of a quiescent c-hub that does not have any active participants in any c-spaces. Because there were no active participants, there were no active conversations in progress. An orderly shutdown removes all state records for the active c-hub instance from the persistent storage.
- *Crash*—Any termination other than an orderly shutdown is a crash. A crash leaves state records in persistent storage, which indicates that the c-hub was active when it terminated.

The goal of recovery is to reconstruct the last known stable run-time image of a c-hub after a crash so that message processing can continue. During startup, if the Hub State indicates that the c-hub crashed, then the start-up process shifts to recovery mode.

The recovery process does not rely solely on the repository data to initiate the c-hub. Instead, the recovery process reads information from persistent storage for the previous c-hub instance. The state records and the message content or location from persistent storage supersede the repository data. After copying the state records and the message content or location and re-creating the corresponding components, the recovery process reads the repository to configure the rest of the c-hub.

If a c-enabler tries to communicate with the c-hub during recovery, the c-hub sends a message to the c-enabler to wait for the conclusion of the recovery process.

Procedure for Configuring Persistence and Recovery

To configure persistence and recovery:

1. Configure the repository as described in "Configuring the Repository" on page 2-5 in Chapter 2, "Setting Up the C-Hub."
2. In the C-Hub Administration Console, set the recovery mode argument for the Startup class.

Attribute	Value
Name	WLCStartup
Class Name	com.bea.b2b.hub.Startup
Arguments	RecoveryMode=ON|OFF
Abort Startup on Failure	TRUE

   The recovery mode is either `ON` or `OFF`. To create and maintain persistent storage, the recovery mode must be `ON`. The default is `OFF`.

3. If you are using the Java Message Service (JMS) queue, in the WebLogic Server Administration Console, set the data source for the JMS queue to the name of the JDBC connection pool. For example:

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

   For complete information, see "Configuring the Java Message Service Queue" on page 2-10 in Chapter 2, "Setting Up the C-Hub."

4. Enable Two phase Commit for the Data Source.
5. Using the WebLogic Administration Console, define the JMSJDBCStore. Set the connection pool to be the same as that for the repository.

6. Configure your JMSServer to use the JMSJDBCStore defined in step 5.

Listing 5-1 Set the JMSJDBCStore and JMSServer to use the same pool as the repository.

```
  <StartupClass
     ClassName="com.bea.b2b.hub.Startup"
     Name="WLCStartup"
     Arguments="RecoveryMode=ON"
     Targets="myserver"
  />
  <JDBCTxDataSource
    JNDIName="WLCHub.DS"
    Name="WLCHub.DS"
    PoolName="repositoryPool"
    EnableTwoPhaseCommit="true"
    Targets="myserver"
/>
  <JMSJDBCStore
    ConnectionPool="repositoryPool"
    PrefixName="hoa"
    Name="JMSWLCStore"
   />
  <JMSServer
   Name="WLCJMSServer"
   Targets="myserver"
   Store="JMSWLCStore"
  />
```

For more information about configuring the c-hub, see Chapter 2, “Setting Up the C-Hub.”

6 Configuring Business Protocols

The following sections describe how to configure and work with business protocols:

- About Business Protocols
- Configuring a Business Protocol
- Working with the RosettaNet Router

About Business Protocols

A business protocol defines how the c-hub processes business messages, including how to read the messages and how to route the messages to the recipients. A business protocol also specifies persistence, retries, and quality of service. A business protocol is specified by a decoder, a router, a filter, an encoder, and a conversation coordinator. For information about decoders, routers, filters, and encoders, see “Message Processing” on page 1-6 in Chapter 1, “Introducing the C-Hub.”

WebLogic Collaborate enables you to use the following business protocols:

- XOCP (the default protocol for WebLogic Collaborate)
- RosettaNet

You can use logic plug-ins to customize and extend these protocols beyond their out-of-the-box functionality. For information about logic plug-ins, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*. For information about assigning logic plug-ins to business protocols, see Chapter 12, "Working with Logic Plug-Ins."

Configuring a Business Protocol

The following sections describe how to configure each type of business protocol that WebLogic Collaborate supports:

- Configuring XOCP
- Configuring RosettaNet

Configuring XOCP

When you create the c-space, specify the XOCP business protocol.

You can use the C-Hub Administration Console or the Bulk Loader to assign a business protocol to a c-space. For information about using these tools, see the following sections:

- "Creating a New C-Space" on page 16-2 in Chapter 16, "Working with C-Spaces."
- "Importing Data into the Repository" on page 3-3 in Chapter 3, "Working with the Bulk Loader."

The following listing is an example of how to assign the XOCP business protocol to a c-space by means of a repository data file which is used by the Bulk Loader.

Listing 6-1 Assigning the XOCP Business Protocol

```
<c-hub name="MyCHub">
    <c-space name="MyCSpace">
        <business-protocol name="XOCP" \
            url="http://localhost/protocol3"/>
    </c-space>
</c-hub>
```

Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol. The `url` attribute for the `business-protocol` XML element specifies this URL.

Warning: A URL for a c-space/business protocol combination can be used only by the c-hub and c-enablers. If customer-supplied software uses one of these URLs, messages will not be processed correctly.

Configuring RosettaNet

To configure RosettaNet:

1. When you create the c-space, specify the RosettaNet business protocol.

   You can use the C-Hub Administration Console or the Bulk Loader to assign a business protocol to a c-space. For information about using these tools, see the following sections:

   - “Creating a New C-Space” on page 16-2 in Chapter 16, “Working with C-Spaces.”
   - “Importing Data into the Repository” on page 3-3 in Chapter 3, “Working with the Bulk Loader.”

   The following listing is an example of how to assign the RosettaNet business protocol to a c-space by means of a repository data file which is used by the Bulk Loader.

Listing 6-2 Assigning the RosettaNet Business Protocol

```
<c-hub name="MyCHub">
    <c-space name="MyCSpace">
        <business-protocol name="RosettaNet" \
            url="http://localhost/protocol1"/>
    </c-space>
</c-hub>
```

Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol. The `url` attribute for the `business-protocol` XML element specifies this URL.

Warning: A URL for a c-space/business protocol combination can be used only by the c-hub and c-enablers. If customer-supplied software uses one of these URLs, messages will not be processed correctly.

2. Create a trading partner protocol and add it for each trading partner that will use the RosettaNet protocol.

   The new trading partner protocol needs to:

   - Reference a RosettaNet business protocol definition.
   - Identify the RosettaNet DUNS numbers for the trading partner that will use the RosettaNet protocol.
   - Specify the URLs that the trading partner will use to access the c-space.

   You can use the C-Hub Administration Console or the Bulk Loader to add a business protocol to a c-space. For information about using these tools, see the following sections:

   - “Assigning a Trading Partner to a TP Protocol” on page 14-10 in Chapter 14, “Working with Trading Partners.”
   - “Importing Data into the Repository” on page 3-3 in Chapter 3, “Working with the Bulk Loader.”

   The following listing is an example of how to define a trading partner protocol for RosettaNet and assign it to trading partners by means of a repository data file which is used by the Bulk Loader.

Listing 6-3 Defining and Assigning a Trading Partner Protocol

```
<c-hub name="MyCHub">
    <trading-partner name="RosettaNetPartner1">
        <trading-partner-protocol
            url="http://localhost:9000/Servlet1"
            business-id="123456789"
            business-id-type="DUNS"
            business-protocol-def-name="RosettaNet"/>
    </trading-partner>
    <trading-partner name="RosettaNetPartner2">
        <trading-partner-protocol
            url="http://localhost:9000/Servlet2"
            business-id="987654321"
            business-id-type="DUNS"
            business-protocol-def-name="RosettaNet"/>
    </trading-partner>
</c-hub>
```

3. Optionally, you can also set the RosettaNet initialization parameters.

   You must use the Bulk Loader to set the RosettaNet initialization parameters. See "Importing Data into the Repository" on page 3-3 in Chapter 3, "Working with the Bulk Loader."

   The following listing is an example of how to set the initialization parameters for RosettaNet by means of a repository data file which is used by the Bulk Loader.

Listing 6-4 Initializing RosettaNet Parameters

```
<c-hub name="MyCHub">
    <business-protocol-def name="RosettaNet">
        <java-class>com.bea.b2b.protocol.rosettanet.RNProtocol</java-class>
        <decoder>RN-Decoder</decoder>
        <router>RN-Router</router>
        <router>RN-Router-Enqueue</router>
        <filter>RN-Filter</filter>
        <encoder>RN-Encoder</encoder>
```

```
        <parameter name="preamble.validate">false</parameter>
        <parameter name="serviceheader.validate">false</parameter>
    </business-protocol-def>
</c-hub>
```

The following table describes the initialization parameters for the RosettaNet protocol.

Table 6-1 RosettaNet Initialization Parameters

Parameter	Description
preamble.validate	Specifies whether or not to perform full XML validation on the preamble. By default, the c-hub performs full validation. To disable validation, specify a value of `false`. Any other value causes the c-hub to use the default value of true. You might want to disable validation in case of performance problems or XML parser problems even though you risk not being able to detect invalid XML content.
serviceheader.validate	Specifies whether or not to perform full XML validation on the service header. By default, the c-hub performs full validation. To disable validation, specify a value of `false`. Any other value causes the c-hub to use the default value of `true`. You might want to disable validation in case of performance problems or XML parser problems even though you risk not being able to detect invalid XML content.

Working with the RosettaNet Router

The following sections describe how to work with the RosettaNet Router:

- Processing Messages
- Processing XML
- Defining a Conversation

Processing Messages

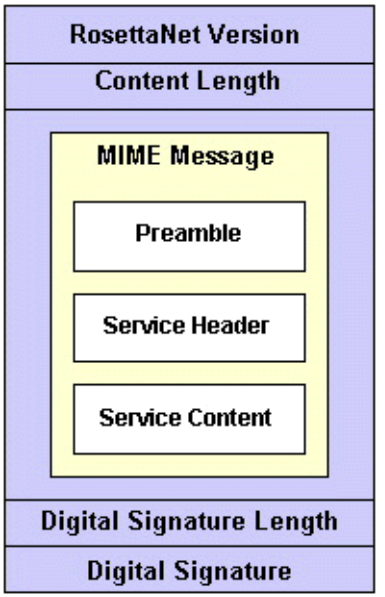
The following sections describe message processing for the RosettaNet router:

- Message Structure
- Message Processing
- Message Routing and Conversation Information
- Security
- RosettaNet Sender

Message Structure

The following figure shows the structure of a RosettaNet message, which is a RosettaNet Object (RNO) based on the RosettaNet Implementation Framework (RNIF) version 1.1. For details about the RNO, see the RNIF at http://www.rosettanet.org.

Figure 6-1 RosettaNet Object



Some sections of the message are binary. For example, the version, the lengths, and the digital signature are binary. The following message sections contain XML:

- Preamble
- Service header
- Service content

Therefore, the message is not a pure XML document, although it includes XML components.

Message Processing

When the c-hub receives a RosettaNet message and sends it to the decoder section of the business protocol for processing:

- The c-hub does not change the message format. The format that the c-hub receives and sends is the one specified by RosettaNet. The c-hub does not perform any additional wrapping or unwrapping of the object, except for the processing described later in this section.
- The c-hub does not validate the digital signature. The receiving trading partner validates the digital signature.
- The c-hub requires read-only access to the message in order to retrieve the information needed for routing. The RosettaNet routing does not modify the message.
- The c-hub requires all the XML components to be available: the preamble, the service header, and the service content. The c-hub requires the first two XML components for message routing. Only the first two XML components are explicitly processed by the c-hub. For information about these components, see "Message Routing and Conversation Information" on page 6-10.

Because of the previously described processing, the c-hub requires that the following information be valid in order to route the message:

- Content type and content length for the HTTP header
- RosettaNet Implementation Framework (RNIF) version
- RNO content length
- MIME multipart structure, including:
  - Boundaries
  - Number of parts
- Valid XML for the preamble
- Valid XML for the service header
- Valid minimal subset of contents within these two XML forms

The message sender must make sure that the information is valid. If the c-hub detects invalid information, the c-hub cannot route the message correctly, logs an error message, and returns a RosettaNet-defined error status to the message sender.

Neither the c-hub nor the RosettaNet router validate the following information in the message:

- Existence of, or correct values for, the content-type for each MIME component.
- Valid XML for the service content. Because the RosettaNet API allows information to be retrieved from this component, the user of the API must be prepared to handle Java exceptions if the service content fails the parser. For information about the RosettaNet API, see the *BEA WebLogic Collaborate Developer Guide*.
- Digital signature.

Message Routing and Conversation Information

The c-hub parses the preamble and service header sections of the RNO to construct routing and conversation information. The following list describes the message fields that must be valid and describes how the c-hub uses these fields. For detailed information about these message fields, see the RNIF at `http://www.rosettanet.org`.

- Preamble
  - **//Preamble/VersionIdentifier:** The RNIF version number. The only RNIF version that WebLogic Collaborate supports is 1.1.
- Service Header
  - **//ServiceHeader/ProcessControl/ProcessIdentity/GlobalProcessIndicatorCode:** The PIP identifier. This typically has a value such as 3A4 and is used as the conversation name. It must be entered as such in the repository.
  - **//ServiceHeader/ProcessControl/ProcessIdentity/VersionIdentifier:** The version of the PIP specification used. This typically has a value such as 1.1 and should not to be confused with the RNIF version number. It is used as the version number of the conversation. It must be entered as such in the repository.

- **//ServiceHeader/ProcessControl/TransactionControl/PartnerRoleRoute/fromRole/PartnerRoleDescription/GlobalPartnerRoleClassificationCode:** The role of the message sender. This typically has a value such as `Buyer` and is used as the role of the message sender in the conversation. It must be entered as such in the repository.
- **//ServiceHeader/ProcessControl/TransactionControl/PartnerRoleRoute/toRole/PartnerRoleDescription/GlobalPartnerRoleClassificationCode:** The role of the recipient. This typically has a value such as `Seller` and is used as the role of the recipient in the conversation. It must be entered as such in the repository.
- **//ServiceHeader/ProcessControl/TransactionControl/<<ActionControl|SignalControl>>/PartnerRoute/fromPartner/PartnerDescription/BusinessDescription/GlobalBusinessIdentifier:** The business identifier of the sender of this message. The path description has an `OR` in the middle, using either the `ActionControl` element or the `SignalControl` element. For the unique business identifier, RosettaNet specifies the use of the 9-digit DUNS number. It must be entered as such in the repository. The business identifier identifies the sending trading partner in the conversation.
- **//ServiceHeader/ProcessControl/TransactionControl/<<ActionControl|SignalControl>>/PartnerRoute/toPartner/PartnerDescription/BusinessDescription/GlobalBusinessIdentifier:** The business identifier of the recipient of this message. It is similar to the `fromPartner` business identifier. The business identifier identifies the recipient trading partner in the conversation.
- **//ServiceHeader/ProcessControl/ProcessIdentity/InstanceIdentifier:** A supposedly unique alphanumeric identifier for this business process, although the RosettaNet specification also states that the initiating partner's business identifier (DUNS number) should be used. These two numbers combine to form the conversation ID, which is used for monitoring and auditing, not for routing.

Note: Neither the c-hub nor RosettaNet enforce the uniqueness of this identifier, especially across multiple vendors. Because the identifier does not play an important role in processing the message, its uniqueness is not critical.

- **//ServiceHeader/ProcessControl/ProcessIdentity/initiatingPartner/GlobalBusinessIdentifier:** The initiating partner's business identifier (DUNS number). As stated earlier, this is used to construct the conversation ID.

From the service content, nothing is required. Depending on the nature of the message, the RosettaNet API either provides access to the following document identifier:

- **//ServiceContent/thisDocumentIdentifier/ProprietaryDocumentIdentifier:** The document identifier. This is the initiating document in the RosettaNet business interaction.

  or it provides access to the following document identifiers:

- **//ServiceContent/thisMessageIdentifier/ProprietaryMessageIdentifier:** The document identifier. This is a follow-on document in the same RosettaNet business interaction.

- **//ServiceContent/receivedDocumentIdentifier/ProprietaryDocumentIdentifier:** The initiating document identifier.

For information about the RosettaNet API, see the *BEA WebLogic Collaborate Developer Guide*.

Security

The following sections describe RosettaNet security:

- SSL
- Digital Signature

For information about WebLogic Collaborate security, see Chapter 4, "Configuring Security."

SSL

When the c-hub decodes a RosettaNet message that it receives by means of SSL, it performs an additional check to verify that the DUNS identifier of the message sender, as retrieved from the message, matches the sender as identified by the certificate. Because the c-hub is acting as a trusted intermediary, this check validates that the sender is who is claimed before the c-hub passes the message to the recipient. If the match fails, the c-hub rejects the message.

Digital Signature

The c-hub processing layers and business protocols do not generate or validate any digital signature associated with the message. Instead, the trading partner clients must validate the digital signature; the c-hub is simply relaying the message. The c-hub validates only the SSL certificate.

The c-hub supports read-only access to a business message. This access ensures that any accompanying digital signature remains valid. You can still introduce logic plug-ins on the c-hub that can validate digital signatures, but you must provide your own third-party public key infrastructure (PKI) library.

RosettaNet Sender

The following sections describe how the c-hub replies to a trading partner that sends a RosettaNet business message:

- RNIF Statement
- Asynchronous Message Transfer
- Validation Disagreement
- Recipient Unavailability

RNIF Statement

Section 3.2.1 of the RNIF states:

"An application that transfers this RosettaNet Object to a remote Web server via a local Web server requests the HTTP protocol to transfer the object as content using the HTTP/1.0 POST request to a target URL. The recipient receives the HTTP request and immediately checks the HTTP headers. If the content-type or transfer encoding is improper, or if the content length fails to match the actual length of the entity body, the recipient returns a 400 (BAD REQUEST) response. If the request is accepted for processing by upper layers in the protocol, a 200 (OK) response will be returned immediately as an acknowledgment of message receipt."

Asynchronous Message Transfer

RNIF 1.1 is a point-to-point protocol and is not designed to interact with a hub. The c-hub delivers messages to recipients by means of asynchronous message transfer. From a RosettaNet perspective, asynchronous message transfer is a "delegated responsibility" model. To route a RosettaNet message and to decide whether or not to reply with an error, the c-hub performs more message-level validation than the recipient needs. After the c-hub accepts a RosettaNet message, the recipient should accept the message without errors. If the recipient notifies the c-hub of a message error, it is too late for the sending trading partner to be notified. The c-hub logs all message responses.

The message persistence capabilities of the c-hub protect the message while it is in the c-hub. The c-hub detects and reports many of the problems that a RosettaNet recipient might detect. If the RosettaNet client starts processing the message and detects errors at the business rule level (message content), the client uses a separate out-of-band error reporting mechanism that is provided by RosettaNet, and this error reporting mechanism is routed through the c-hub like any other RosettaNet message.

Validation Disagreement

If the c-hub and the recipient disagree about the validation:

- If the c-hub rejects a message as invalid that the recipient would accept, then you must determine which party is non-compliant (the sending trading partner, the c-hub, or the recipient) and take appropriate action. For example, if the c-hub is non-compliant, then you might need to modify the RosettaNet protocol or the initialization parameters.
- If the recipient rejects a message that the c-hub accepted, you must determine the non-compliant party. However, under these circumstances the recipient responds to the c-hub with a BAD REQUEST message, which the c-hub logs. The c-hub cannot initiate RosettaNet error responses to the sending trading partner for a number of reasons, including digital signature implications and lack of support for hub-based error-reporting in RNIF.

Recipient Unavailability

The c-hub can accept a RosettaNet message from a trading partner and return an HTTP 200 to the sending trading partner. However, the recipient might not be available when the c-hub is prepared to deliver the message. In this case, the message will be lost. By

using retries and timeouts, the RosettaNet client protocol is robust enough to handle lost messages and no response situations. The c-hub logs the delivery status of all RosettaNet messages.

Processing XML

The RosettaNet router uses information in the preamble and service header. The c-hub uses a validating XML parser to parse the preamble and service header. If the c-hub detects an error, it logs the error and returns the status to the sending trading partner.

The RosettaNet router does not use information in the service content section. The service content is the business-specific section of the message and is handled by the receiving client. The c-hub can provide the service content information for monitoring purposes. If needed, the c-hub parses the service content as a "well-formed" (non-validating) XML document in order to retrieve fields. If the service content is not well-formed, the c-hub logs an error message.

For each XML section, the DOM document can be retrieved. You can then use either the DOM or XPath API to access any element of the document.

Logic plug-ins do not prevent you from parsing XML and processing the service content. Logic plug-ins enable you to filter or monitor messages based on the service content section of the message. You are responsible for the parsing, however, in order to do the filtering.

Defining a Conversation

Conversations are defined in the repository. To define a conversation, do one of the following:

- Use the C-Hub Administration Console as described in Chapter 15, "Setting Up Conversations."
- Use the Bulk Loader to import data into the repository as described the "Importing Data into the Repository" section in Chapter 3, "Working with the Bulk Loader."

The following listing is an example of a conversation definition in a repository data file.

Listing 6-5 Example Conversation Definition Values in a Repository Data File

```
<conversation-def
    name="3A4"
    version="1.1"
    default-durability="<persistent|nonpersistent>"
    default-conversation-timeout="<timeout-in-seconds>">
    <role
        name="Buyer">
    </role>
    <role
        name="Seller">
    </role>
</conversation-def>
```

The DTD for the repository data file is `WLCHub.dtd`, which is in the `bulkloader` subdirectory of your WebLogic Collaborate installation directory. The `WLCHub.dtd` file describes the format of and values in a conversation definition.

When you use the C-Hub Administration Console to monitor RosettaNet conversations, the console displays a conversation ID that the c-hub constructs from a RosettaNet message:

<protocol-name>:*<PIP-id>*:*<PIP-version>*:*<initiator-DUNS>*:*<process-instance-identifier>*

In this conversation ID:

- *protocol-name* is the protocol name as defined in the repository. For example, "RosettaNet."
- *PIP-id* is the PIP identifier associated with the message. For example, "3A4."
- *PIP-version* is the version of the PIP being used. For example, "1.1."

- *`initiator-DUNS`* is the DUNS number of the trading partner who initiated the RosettaNet dialog. For example, “123456789.”
- *`process-instance-identifier`* is the process instance identifier value obtained from the RosettaNet service header. For example, “15902235777701132.”

RosettaNet handles conversation management externally. Therefore, this conversation definition does not provide a conversation termination mechanism for terminating RosettaNet clients. The conversation definition provides a time-out mechanism to remove RosettaNet entries. The c-hub uses this mechanism only to remove entries from the monitoring mode of the C-Hub Administration Console. Because the c-hub registers these entries in real time from external RosettaNet clients, new messages succeed and are displayed in the monitoring mode.

The conversation definition also enables you to specify conversation persistence. If the conversation persistence field is set, the RosettaNet protocol processes the messages it passes through the c-hub in a persistent fashion. If the c-hub crashes, the c-hub recovers these messages and continues processing them.

7 Routing and Filtering XOCP Business Messages

The following sections describe how to use routing, filtering, and XPath expressions to control the flow of XOCP business messages exchanged among trading partners in a c-space:

- Run-Time Message Processing
- Working with Message-Context Documents
- Working with XPath Expressions

Run-Time Message Processing

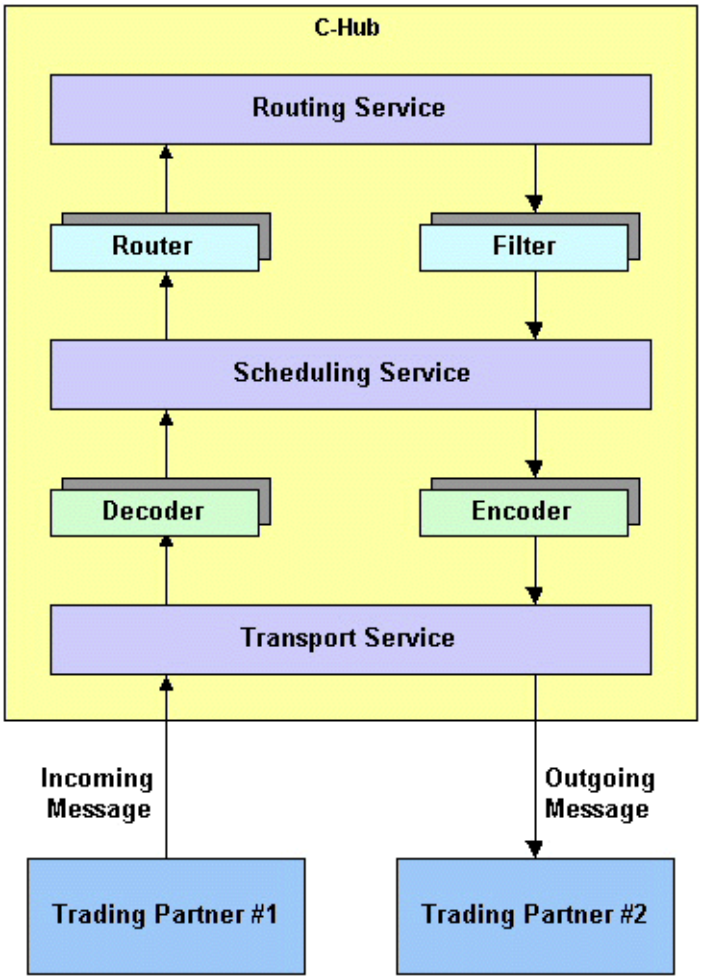
BEA WebLogic Collaborate uses the XOCP router and XOCP filter to direct the flow of XOCP business messages through the c-hub to trading partners in a c-space as follows:

- After a trading partner sends an XOCP business message to the c-hub, the **XOCP router** determines the trading partners to which the message will be sent. The XOCP router is on the "send" side of the c-hub message processing and determines the recipients to which the sending trading partner intends to send the message.

- Before the c-hub sends the business message to a recipient trading partner, the **XOCP filter** determines whether or not the trading partner should receive it. The XOCP filter is on the "receive" side of the c-hub message processing and can prevent a specific trading partner from receiving a specific business message.

The following figure provides an overview of how a c-hub processes a message.

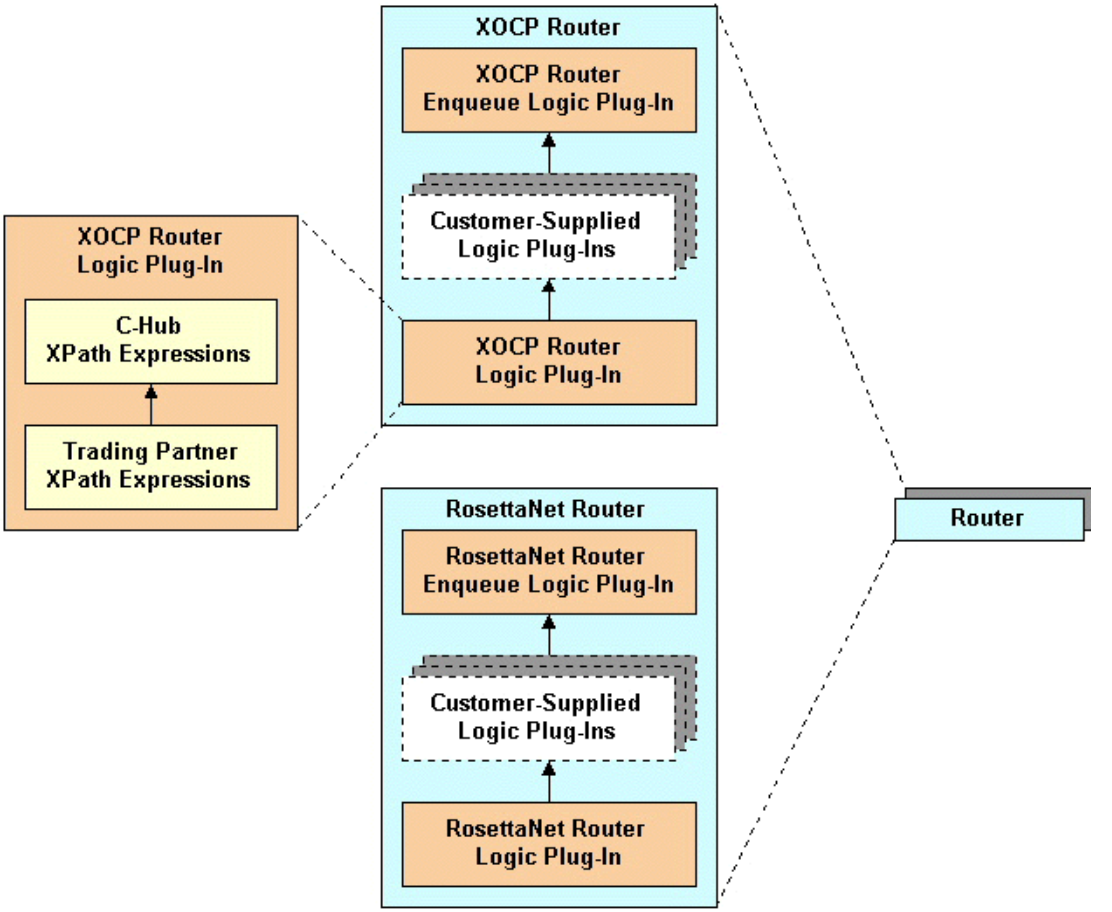
Figure 7-1 Overview of Message Processing



The following figure provides a detailed look at the router. The router consists of a protocol-specific router for each business protocol that WebLogic Collaborate supports.

Note: For completeness, RosettaNet is included in the diagram even though RosettaNet is not part of XOCP message processing.

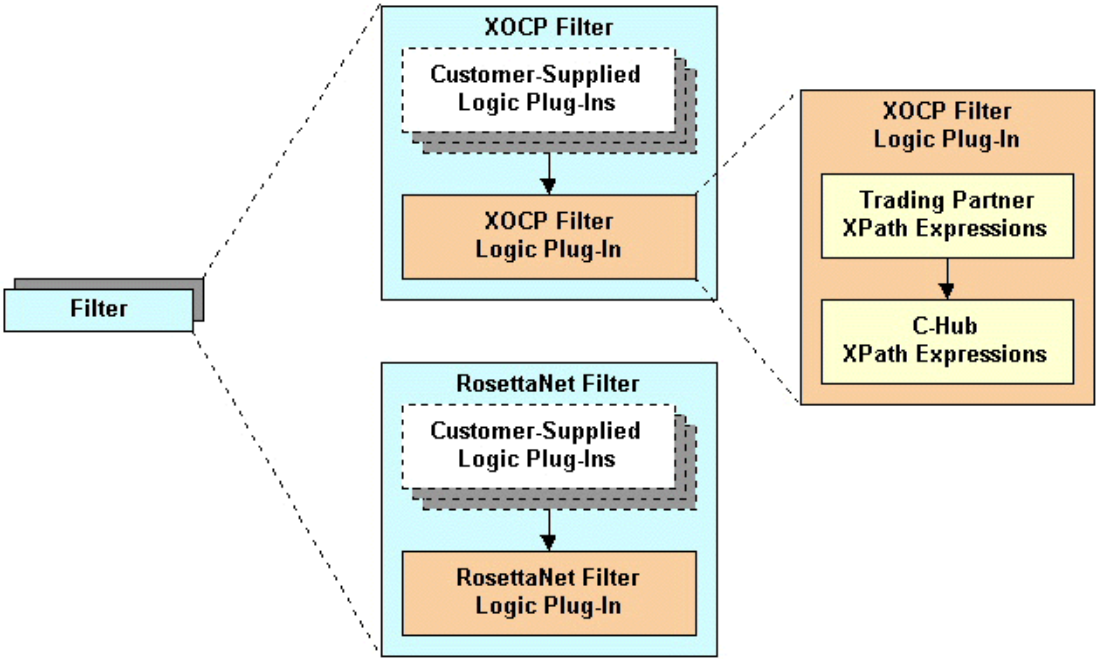
Figure 7-2 Router



The following figure provides a detailed look at the filter. The filter consists of a protocol-specific filter for each business protocol that WebLogic Collaborate supports.

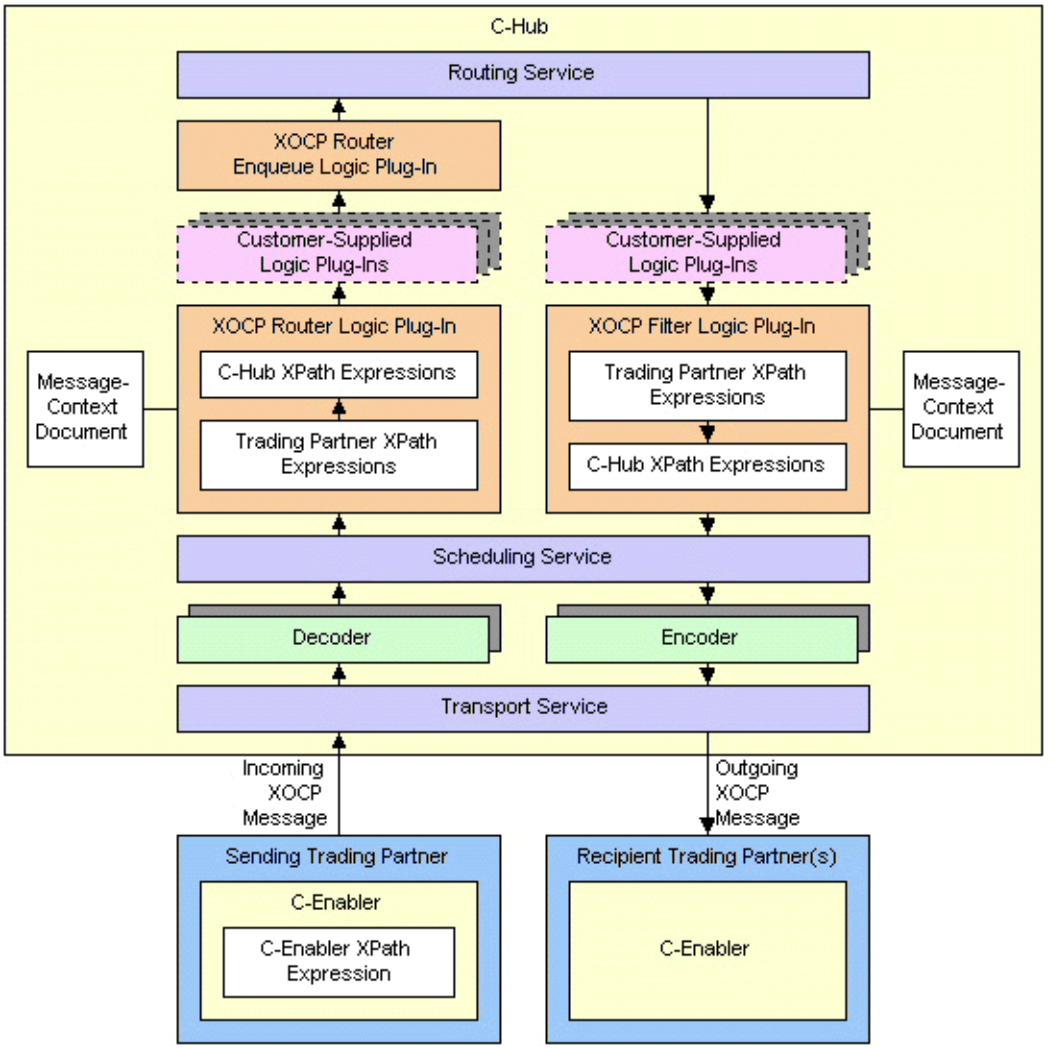
Note: For completeness, RosettaNet is included in the diagram even though RosettaNet is not part of XOCP message processing.

Figure 7-3 Filter



The following figure provides a detailed look at how a c-hub processes an XOCP business message.

Figure 7-4 XOCP Message Processing



The following sections explain how the "send" and "receive" sides of the c-hub process an XOCP business message:

- The Send Side
- The Receive Side

The Send Side

The following sections describe the components in the "send" side of the c-hub and explain how they process an XOCP business message:

- C-Enabler XPath Expression
- Transport Service
- Decoder
- Scheduling Service
- XOCP Router
- Routing Service

C-Enabler XPath Expression

When sending an XOCP business message, the c-enabler for the sending trading partner can specify a c-enabler XPath expression that defines the intended recipients for the business message. The c-enabler XPath expression is defined in a WebLogic Process Integrator workflow or in a c-enabler application. For more information about c-enabler XPath expressions, see "Creating C-Enabler XPath Expressions" on page 7-21.

Transport Service

The transport service reads the incoming XOCP business message and does the following:

1. Wraps the message in a message envelope to expedite processing as it travels through the c-hub.
2. Forwards the message to the appropriate decoder based on the business protocol, such as XOCP or RosettaNet. The URL on which the transport service receives the message identifies the protocol and the c-space. Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol.

   Warning: A URL for a c-space/business protocol combination can be used only by the c-hub and c-enablers. If customer-supplied software uses one of these URLs, messages will not be processed correctly.

   For information about business protocols, see Chapter 6, "Configuring Business Protocols."

Decoder

The decoder does the following:

1. Processes the protocol-specific headers.
2. Identifies the sending trading partner.
3. Enlists the sending trading partner in a conversation.
4. Prepares a reply to return to the sender.
5. Forwards the message to the scheduling service.

Scheduling Service

The scheduling service enqueues the message to store it for subsequent retrieval and forwards the message to the XOCP router.

XOCP Router

The XOCP router is a chain of logic plug-ins that specifies the recipients for the XOCP business message. Each logic plug-in can add trading partners to or remove trading partners from the set of recipient trading partners.

The order of the logic plug-ins in the XOCP router is:

1. XOCP router logic plug-in—provided by WebLogic Collaborate
2. Customer-supplied logic plug-ins—optional logic plug-ins that you can create
3. XOCP router enqueue logic plug-in—provided by WebLogic Collaborate

The following sections describe these logic plug-ins.

XOCP Router Logic Plug-In

The XOCP router logic plug-in does the following:

1. Creates a message-context document.

   A message-context document is an XML document that the XOCP router logic plug-in generates from the XOCP business message and associated information in the repository. The message-context document describes header and payload information about the XOCP business message, such as the c-hub, large message information, c-space, business protocol, conversation, sending trading partner, receiving trading partners, message definition, and document definition. The XOCP router logic plug-in uses XPath expressions to evaluate the message-context document. For more information about message-context documents, see “Working with Message-Context Documents” on page 7-14.

2. Evaluates the message-context document against the XPath routing expressions, which can refer to values in the message-context document. This evaluation results in a set of trading partners that are targeted to receive the XOCP business message.

   The XOCP router logic plug-in uses the XPath routing expressions in the following order:

   a. C-enabler XPath expression.

      For information about c-enabler XPath expressions, see “C-Enabler XPath Expression” on page 7-6 and “Creating C-Enabler XPath Expressions” on page 7-21.

b. Sequence of trading partner XPath routing expressions.

   These XPath routing expressions are defined in the repository and are defined for the sending trading partner. Each trading partner XPath routing expression applies to all XOCP business messages that the c-hub receives from a particular sending trading partner. Each sending trading partner can have multiple trading partner XPath routing expressions.

   Each trading partner XPath routing expression can examine different parts of the message-context document and select a different set of recipient trading partners. The trading partners produced by each expression can either replace the previously generated set of recipient trading partners or add to the current set.

c. Sequence of c-hub XPath routing expressions.

   These XPath routing expressions are defined in the repository and are defined for the c-hub. Each c-hub XPath routing expression applies to all XOCP business messages that the c-hub receives from all sending trading partners. The c-hub can have multiple c-hub XPath routing expressions.

   As with trading partner XPath routing expressions, each c-hub XPath routing expression can examine different parts of the message-context document and select a different set of recipient trading partners. The trading partners produced by each expression can either replace the previously generated set of recipient trading partners or add to the current set.

You can add XPath expressions to the repository for use by the XOCP router logic plug-in. For information about XPath expressions, see "Working with XPath Expressions" on page 7-17.

3. Discards the message-context document.
4. If the set of recipient trading partners is empty, then the XOCP router logic plug-in does not forward the message to the next component in the c-hub. Otherwise, the c-hub continues to process the message.

Customer-Supplied Logic Plug-Ins

You can create logic plug-ins and add them to the XOCP router. If you create a new logic plug-in, you must add it to the chain after the XOCP router logic plug-in and before the XOCP router enqueue logic plug-in. The order of the logic plug-ins in the XOCP router chain is specified in the XOCP business protocol definition.

A customer-supplied logic plug-in does not have to provide router functionality to be part of the XOCP router. For example, a customer-supplied logic plug-in can provide billing functionality by keeping track of the number of message sent by a particular sending trading partner and then billing the trading partner for those messages. Even when a customer-supplied logic plug-in does not provide routing or filtering functionality, it can be added only to the XOCP router or the XOCP filter. For more information about logic plug-ins, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*.

If the set of recipient trading partners is empty after the processing performed by a customer-supplied router logic plug-in, then the customer-supplied router logic plug-in does not forward the message to the next component in the c-hub. Otherwise, the c-hub continues to process the message.

XOCP Router Enqueue Logic Plug-In

The XOCP router enqueue logic plug-in does the following:

1. Enqueues the XOCP business message along with the intended recipients.
2. Forwards the message to the routing service.

Routing Service

The routing service does the following:

1. Performs the final validation of the message recipients.
2. Creates a separate message envelope, along with a copy of the message content, for each validated recipient trading partner.
3. Stores each copy of the message for delivery.
4. Forwards each copy of the message to the XOCP filter.

The Receive Side

The following sections describe the components in the “receive” side of the c-hub and explain how they process an XOCP business message:

- XOCP Filter
- Scheduling Service
- Encoder
- Transport Service

XOCP Filter

The XOCP filter is a chain of logic plug-ins that determines whether or not to send an XOCP business message to an intended recipient. These logic plug-ins are evaluated after the XOCP router logic plug-ins and can modify or override the XOCP router results. Each logic plug-in can determine not to send the message.

The order of the logic plug-ins in the XOCP filter is:

1. Customer-supplied logic plug-ins—optional logic plug-ins that you can create
2. XOCP filter logic plug-in—provided by WebLogic Collaborate

The following sections describe these logic plug-ins.

XOCP Filter Logic Plug-In

The XOCP filter logic plug-in does the following:

1. Creates a message-context document.

   A message-context document is an XML document that the XOCP filter logic plug-in generates from the XOCP business message and associated information in the repository. The message-context document describes header and payload information about the XOCP business message, such as the c-hub, large message information, c-space, business protocol, conversation, sending trading partner, receiving trading partners, message definition, and document definition. The XOCP filter logic plug-in uses XPath expressions to evaluate the message-context document. For more information about message-context documents, see “Working with Message-Context Documents” on page 7-14.

2. Evaluates the message-context document against the XPath filtering expressions, which can refer to values in the message-context document. This evaluation determines whether or not to send the message to the intended recipient.

   The XOCP filter logic plug-in uses the XPath filtering expressions in the following order:

   a. Sequence of trading partner XPath filtering expressions.

      These XPath filtering expressions are defined in the repository and are defined for the recipient trading partner. Each trading partner XPath filtering expression applies to all XOCP business messages that the c-hub receives for a particular recipient trading partner. Each recipient trading partner can have multiple trading partner XPath filtering expressions.

      Each trading partner XPath filtering expression can examine different parts of the message-context document and return a Boolean result that indicates acceptance or rejection of the message.

   b. Sequence of c-hub XPath filtering expressions.

      These XPath filtering expressions are defined in the repository and are defined for the c-hub. Each c-hub XPath filtering expression applies to all XOCP business messages that the c-hub receives for all recipient trading partners. The c-hub can have multiple c-hub XPath filtering expressions.

      As with trading partner XPath filtering expressions, each c-hub XPath filtering expression can examine different parts of the message-context document and return a Boolean result that indicates acceptance or rejection of the message.

   You can add XPath expressions to the repository for use by the XOCP filter logic plug-in. For information about XPath expressions, see “Working with XPath Expressions” on page 7-17.

3. Discards the message-context document.

4. If the XOCP filter logic plug-in cancels delivery of the XOCP business message to the intended recipient, then the XOCP filter logic plug-in does not forward the message to the next component in the c-hub. Otherwise, the c-hub continues to process the message.

Customer-Supplied Logic Plug-Ins

You can create logic plug-ins and add them to the XOCP filter. If you create a new logic plug-in, you must add it to the chain before the XOCP filter logic plug-in. The order of the logic plug-ins in the XOCP filter chain is specified in the XOCP business protocol definition.

A customer-supplied logic plug-in does not have to provide filter functionality to be part of the XOCP filter. For example, a customer-supplied logic plug-in can provide sampling functionality by keeping track of the types of messages sent to a particular recipient trading partner. Even when a customer-supplied logic plug-in does not provide routing or filtering functionality, it can be added only to the XOCP router or the XOCP filter. For more information about logic plug-ins, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*.

If the customer-supplied logic plug-ins cancels delivery of the XOCP business message to the intended recipient, then the customer-supplied filter logic plug-in does not forward the message to the next component in the c-hub. Otherwise, the c-hub continues to process the message.

Scheduling Service

The scheduling service does the following:

1. Performs additional internal operations related to quality of service issues and conversation management. For information about quality of service, see the *BEA WebLogic Collaborate Developer Guide*.
2. Forwards the message to the encoder.

Encoder

The encoder transforms the message as necessary to support the business protocol and forwards the message to the transport service.

Transport Service

The transport service sends the message to the recipient.

Working with Message-Context Documents

For information about how message-context documents are created and used, see "XOCP Router Logic Plug-In" on page 7-8 and "XOCP Filter Logic Plug-In" on page 7-11.

This section describes message-context documents:

- DTD for the Message-Context Document
- Sample Message-Context Document and XPath Expressions

DTD for the Message-Context Document

The following listing is the Document Type Definition (DTD) for the message-context document.

Listing 7-1 Document Type Definition for Message-Context Document

```
<!--Copyright (c) 2000 BEA Systems, Inc. -->
<!--All rights reserved                  -->

<!-- This DTD describes the message context document for
    XPath routing expressions and XPath filtering expressions -->

<!ELEMENT c-hub (c-space, conversation, sender, trading-partner+,
    message)>
<!ATTLIST c-hub context (message-router | trading-partner-router | hub-router |
trading-partner-filter | hub-filter) #REQUIRED>
<!ATTLIST c-hub name CDATA #IMPLIED>
<!ATTLIST c-hub large-msg-support-on CDATA #IMPLIED>
<!ATTLIST c-hub large-msg-min-size CDATA #IMPLIED>
<!ATTLIST c-hub large-msg-location CDATA #IMPLIED>

<!ELEMENT c-space (business-protocol)>
<!ATTLIST c-space name CDATA #REQUIRED>

<!ELEMENT business-protocol EMPTY>
<!ATTLIST business-protocol name CDATA #REQUIRED>
```

```
<!ATTLIST business-protocol url CDATA #REQUIRED>

<!ELEMENT conversation EMPTY>
<!ATTLIST conversation name CDATA #REQUIRED>
<!ATTLIST conversation version CDATA #REQUIRED>
<!ATTLIST conversation default-durability CDATA #IMPLIED>
<!ATTLIST conversation default-conversation-timeout
    CDATA #IMPLIED>
<!ATTLIST conversation sender-role CDATA #REQUIRED>
<!ATTLIST conversation receiver-role CDATA #REQUIRED>

<!-- A sender is a trading-partner that has sent a message from a role in a
conversation. -->
<!ELEMENT sender (trading-partner)>

<!-- A Trading Partner represents an entity such as a company that sends or
receives messages. -->
<!ELEMENT trading-partner (address, extended-property-set*)>
<!ATTLIST trading-partner email CDATA #IMPLIED>
<!ATTLIST trading-partner fax CDATA #IMPLIED>
<!ATTLIST trading-partner name ID #REQUIRED>
<!ATTLIST trading-partner phone CDATA #IMPLIED>
<!ATTLIST trading-partner certificate-field CDATA #IMPLIED>
<!ATTLIST trading-partner certificate-field-value CDATA #IMPLIED>
<!ATTLIST trading-partner user-name CDATA #IMPLIED>

<!ELEMENT address ANY>

<!ELEMENT extended-property-set ANY>
<!ATTLIST extended-property-set name CDATA #REQUIRED>

<!-- A message is a multipart payload sent between trading partners -->
<!ELEMENT message (message-part+)>
<!ATTLIST message message-def-name CDATA #REQUIRED>

<!ELEMENT message-part ANY>
<!ATTLIST message-part document-def-name CDATA #IMPLIED>
<!ATTLIST message-part content-type CDATA #REQUIRED>
<!-- A message part may include or exclude the subelements based on admin settings
or mime-type -->
<!ATTLIST message-part include (yes | no) #REQUIRED>
```

Sample Message-Context Document and XPath Expressions

The following listing shows a sample message-context document generated by the XOCP router logic plug-in for a business message.

Listing 7-2 Sample Message-Context Document

```
<c-hub name="CHub1">
    <c-space name="Cspace1">
        <business-protocol name="XOCP1" url="http://myserver/Cspace1/XOCP1"/>
    </c-space>
    <conversation name="aConv" version="1.0">
        <sender role="buyer"/>
        <receiver role="seller"/>
    </conversation>
    <sender>
        <trading-partner name="Partner1"
            email="messages@partner1.com"
            phone="123-456-7890"
            fax="123-456-7899">
            <address>123 Main Street, My Town, CA 95131</address>
            <extended-property-set name="Partner1 contact">
                <business-contact>John Doe</business-contact>
                <phone type="work">123-456-7898</phone>
                <phone type="cell">123-456-7897</phone>
                <city>San Jose</city>
                <state>California</state>
            </extended-property-set>
        </trading-partner>
    </sender>

    <!-- List of target trading partners extracted from the repository -->
    <trading-partner name="Partner2"
        email="messages@partner2.com"
        phone="123-456-7896">
        <address>456 Another Street, Any Town, CA 95136</address>
        <extended-property-set name="Partner2 contact">
            <business-contact>Jane Smith</business-contact>
            <phone type="work">123-456-7895</phone>
            <phone type="cell">123-456-7894</phone>
            <city>San Jose</city>
            <state>California</state>
        </extended-property-set>
```

```
    </trading-partner>

    <message message-def-name="Product Query">
        <message-part document-def-name="ProductQueryDefinition"
            content-type="text/xml" include="yes">
            <ProductPriceAndAvailabilityQuery>
        </message-part>
    </message>
</c-hub>
```

For this XOCP business message, XPath routing expressions in the XOCP router logic plug-in could refer to any of the elements in this XML document, including extended properties and message parts. The following XPath routing expression routes messages to trading partners that have extended properties specifying that they are located in San Jose, California:

```
/c-hub/trading-partner[extended-property-set[city='San Jose' AND
state='California']]
```

Working with XPath Expressions

This section describes XPath expressions and how to create them:

- About XPath Expressions
- Creating C-Enabler XPath Expressions
- Creating Trading Partner XPath Expressions
- Creating C-Hub XPath Expressions

About XPath Expressions

XPath is the XML path language that is defined by the World Wide Web Consortium. The XOCP router logic plug-in and the XOCP filter logic plug-in use XPath expressions to evaluate message-context documents. You can add XPath expressions to the repository for use by the XOCP router logic plug-in and the XOCP filter logic plug-in.

XPath expressions in the XOCP router logic plug-in and XOCP filter logic plug-in perform the following functions:

- An XPath routing expression uses the XPath syntax to select a set of trading partners from the message-context document. These trading partners are the intended recipients of the XOCP business message. Each XPath routing expression must evaluate to a set of trading partners.

  In the XOCP router logic plug-in, XPath expressions specify the business criteria for message distribution. For example, a buyer can use an XPath routing expression to send bid requests to all sellers in a particular area code or to sellers that can handle large orders.

- An XPath filtering expression uses the XPath syntax to return a Boolean result that indicates acceptance or rejection of the message. Each XPath filtering expression must evaluate to a Boolean `true` or `false` result.

  In the XOCP filter logic plug-in, XPath expressions determine whether or not the c-hub sends a particular business message to a particular trading partner. An XPath filtering expression in the XOCP filter logic plug-in acts as a gatekeeper that filters out unwanted business messages for a receiving trading partner. For example, a seller can specify an XPath filtering expression that accepts bid requests with a minimum total order of $10,000 or a maximum quantity of 100,000 units. Similarly, a buyer can specify an XPath filtering expression that ignores bids from a certain seller or bids that exceed a maximum per unit cost.

The following table provides an overview of the various types of XPath expressions.

Table 7-1 Overview of Types of XPath Expressions

Type of XPath Expression	XOCP Router Logic Plug-In	XOCP Filter Logic Plug-In
C-enabler	Evaluated: **first** # of XPath expressions: one Defined by: c-enabler (in workflow or application) Purpose: defines recipients Applies to: XOCP business messages from the sending c-enabler	Not applicable
Trading partner	Evaluated: **second** # of XPath expressions: one or more Defined in: repository (via C-Hub Administration Console or Bulk Loader) Purpose: adds and removes recipients Applies to: all XOCP business messages from the sending trading partner	Evaluated: **fourth** # of XPath expressions: one or more Defined in: repository (via C-Hub Administration Console or Bulk Loader) Purpose: determines whether or not to send the message to the recipient Applies to: all XOCP business messages to the recipient trading partner
C-hub	Evaluated: **third** # of XPath expressions: one or more Defined in: repository (via C-Hub Administration Console or Bulk Loader) Purpose: adds and removes recipients Applies to: all XOCP business messages from all sending trading partners	Evaluated: **fifth** # of XPath expressions: one or more Defined in: repository (via C-Hub Administration Console or Bulk Loader) Purpose: determines whether or not to send the message to the recipient Applies to: all XOCP business messages to all recipient trading partners

In the XOCP router logic plug-in, each XPath routing expression can examine different parts of the message-context document and select a different set of recipient trading partners. The trading partners produced by each expression can either replace the previously generated set of recipient trading partners or add to the current set.

The following table steps through an example that shows how XPath routing expressions can be used.

Table 7-2 Example for XPath Routing Expressions

XPath Expression	Resulting Set of Recipient Trading Partners
1. The c-enabler XPath expression selects trading partners A and B.	A, B
2. A trading partner XPath routing expression adds trading partner C.	A, B, C
3. A subsequent trading partner XPath routing expression replaces all previously selected trading partners with trading partner D.	D
4. A c-hub XPath routing expression adds trading partners B and F.	D, B, F
5. A subsequent c-hub XPath routing expression remove trading partner F.	D, B

In the XOCP filter logic plug-in, each XPath filtering expression can examine different parts of the message-context document to determine whether or not to forward the message to the recipient trading partner. Each XPath filtering expression can return `true` or `false` using different selection criteria. After an XPath filtering expression returns `false`, the message is blocked from further evaluation and is not sent to the intended recipient.

An XPath expression can refer to the following kinds of information:

- Trading partner attributes, including:
  - Standard attributes, such the trading partner name or a postal code
  - Extended attributes, which are custom attributes defined by the c-hub administrator
- Message payload information, such as the type of business document, a purchase order number, or an invoice amount

Creating C-Enabler XPath Expressions

When sending an XOCP business message, the c-enabler for the sending trading partner can specify a c-enabler XPath expression that defines the intended recipients for the business message. The c-enabler XPath expression is defined in a WebLogic Process Integrator workflow or in a c-enabler application. This XPath expression selects a subset of `<trading-partner>` nodes from the message-context XML document that the XOCP router logic plug-in will generate.

The sending c-enabler defines this XPath expression and sends it to the c-hub along with the message. The c-enabler defines an XPath expression as follows:

- If the c-enabler uses a workflow to exchange business messages, the XPath expression is defined in the workflow definition template and applied when the c-enabler sends the message to the c-hub. Use the Send Business Message dialog to define the XPath expression. For more information, see Using Workflows to Exchange Business Messages in the *BEA WebLogic Collaborate Developer Guide*.
- If the c-enabler uses a c-enabler application to exchange business messages, the XPath expression is defined in the c-enabler application when it sends the message to the c-hub. Call the `setExpression` method on the `com.bea.b2b.protocol.messaging.Message` instance, passing the XPath expression as the parameter. For more information, see Using XOCP C-Enabler Applications to Exchange Business Messages in the *BEA WebLogic Collaborate Developer Guide*.

  Note: In many cases, a c-enabler application sends a business message to a single, known trading partner; for example, to reply to a request from that trading partner. In this case, a c-enabler application can bypass the evaluation of XPath expressions in the XOCP router logic plug-in by specifying a trading partner name instead of an XPath expression. To specify a trading partner name, call the `setRecipient` method instead of `setExpression` on the `com.bea.b2b.protocol.messaging.Message` instance.

Creating Trading Partner XPath Expressions

A trading partner XPath expression is an XPath expression that is defined for a trading partner. For routing, a trading partner XPath expression is used by the XOCP router logic plug-in and is defined for the sending trading partner. For filtering, a trading partner XPath expression is used by the XOCP filter logic plug-in and is defined for the receiving trading partner.

Trading partner XPath expressions are defined in the repository. You can use the following tools to create trading partner XPath expressions for the XOCP router logic plug-in and the XOCP filter logic plug-in:

- Bulk Loader as described in "Importing Data into the Repository" on page 3-3 in Chapter 3, "Working with the Bulk Loader." The format for an XPath expression in a repository data file is:

  ```
  <xpath-exp expression="XPath_expression" type="type"/>
  ```

  For more information about XPath syntax and usage, see *The XML Path Language Specification*, published by the World Wide Web Consortium, at the following URL: `http://www.w3.org/TR/xpath.html`.

- C-Hub Administration Console as described in "Defining XOCP Filters and Routers for a Trading Partner" on page 14-5 in Chapter 14, "Working with Trading Partners."

The following table describes the properties that you set when using the C-Hub Administration Console to define XPath expressions for trading partners and the c-hub.

Table 7-3 Properties for XPath Expressions

Component	Description
XPath Expression	XPath routing or filtering expression as previously described.
Type	Flag that specifies whether the results of evaluating the XPath expression append or replace the results of the evaluations of the previous XPath expressions.
Use Payload	Flag that specifies whether or not the XPath expression refers to information in the payload part of the business message. If the XPath expression references the payload, then the c-hub will expand the payload during message processing. **Note:** For large messages, expanding the payload could have a significant performance impact. If the XPath expression does not reference the payload, it is recommended that you should not select this option. The XPath expression can refer only to elements in an XML document, not to bytes in an attachment.

For example, a trading partner might want to route requests to trading partners that are located in California. To do this, the trading partner can use the detail screen on the Trading Partners tab in the C-Hub Administration Console to create the following XPath expression for the XOCP router logic plug-in:

```
/c-hub/trading-partner[extended-property-set/state='California']
```

For another example, a trading partner might want to ignore bid requests for quantities less than 100. To do this, the trading partner can use the detail screen on the Trading Partners tab in the C-Hub Administration Console to create the following XPath expression for the XOCP filter logic plug-in:

```
(/c-hub/message/@message-def-name='Product Bid' AND
/c-hub/message/message-part[1]/quantity >= 100])
```

Creating C-Hub XPath Expressions

A c-hub XPath expression is an XPath expression that is defined for a business protocol. For routing, a c-hub XPath expression is used by the XOCP router logic plug-in and is defined for all the XOCP business messages that pass through the c-hub. For filtering, a c-hub XPath expression is used by the XOCP filter logic plug-in and is defined for all the XOCP business messages that pass through the c-hub.

C-hub XPath expressions are defined in the repository. You can use the following tools to create c-hub XPath expressions for the XOCP router logic plug-in and the XOCP filter logic plug-in:

- Bulk Loader as described in "Importing Data into the Repository" on page 3-3 in Chapter 3, "Working with the Bulk Loader." The format for an XPath expression in a repository data file is:

  `<xpath-exp expression="`*`XPath_expression`*`" type="`*`type`*`"/>`

  For more information about XPath syntax and usage, see *The XML Path Language Specification*, published by the World Wide Web Consortium, at the following URL: `http://www.w3.org/TR/xpath.html`.

- C-Hub Administration Console as described in "Assigning Logic Plug-Ins to Business Protocols Definitions" on page 13-2 in Chapter 13, "Working with Business Protocol Definitions."

  Table 7-3 describes the properties that you set when using the C-Hub Administration Console to define an XPath expression.

For example, a c-hub administrator might want to filter messages for trading partners that are shippers so that they receive only shipping requests, while all other types of trading partners receive all messages. To do this, the c-hub administrator can use the Business Protocol Definitions tab in the C-Hub Administration Console to create the following XPath expression for the XOCP filter logic plug-in:

```
(/c-hub/message/@message-def-name='Shipping Request' AND
/c-hub/trading-partner/extended-property-set/business='shipper') OR
(/c-hub/trading-partner/extended-property-set/business!='shipper')
```

Part II Using the C-Hub Administration Console

This section is a copy of the C-Hub Administration Console Online Help. To start the C-Hub Administration Console, see “Configuring and Running the C-Hub Administration Console” on page 2-11.

CHAPTER

8 Getting Started with C-Hub Administration

The following sections provide an overview of the C-Hub Administration Console and explain how to log on and get started using the console:

- Logging On to the C-Hub Administration Console
- Starting the C-Hub
- Stopping the C-Hub
- Logging Off the C-Hub Administration Console
- Overview of the C-Hub Administration Console
- What's Next?

For more information on c-hubs and c-hub administration, see *BEA WebLogic Collaborate Getting Started* and the *BEA WebLogic Collaborate Developer Guide*.

Logging On to the C-Hub Administration Console

When you start the C-Hub Administration Console, you get a logon screen.

Figure 8-1 C-Hub Logon Screen

Before you can do any c-hub administration tasks (monitoring or configuration), you must log on to the system.

Table 8-1 Logon Fields

Field	Description
User Name	Enter your user name as defined in the administration security section of the WebLogic Server Administration Console. For more information on this, see Chapter 2, "Setting Up the C-Hub," and Chapter 4, "Configuring Security."
User Password	Enter your user password.

Note: User names and passwords can be limited to monitoring only, or granted both monitoring and configuration privileges.

When you have filled in the fields, click **Submit** to log on (or **Reset** to start over).

Submit checks your user credentials against the current WebLogic Server realm. If your user credentials are valid, the system checks to determine which administration facility (configuration and/or monitoring) you are allowed to access.

If you are denied access, the same screen is displayed and a message informs you that access is denied.

If you are allowed access, a similar screen is displayed without the login fields, as shown in the following figure.

Figure 8-2 C-Hub Administration Console Starting Screen after Login

Once you are logged in, you can do any of the following:

- Click **Configuration** to begin configuring a c-hub and the objects on it. (You can do this only if your user name and password carry configuration privileges.)
- Click **Monitoring** to begin viewing reports on a running system.
- Click **Help** to get help information on the C-Hub Administration Console.
- Click **Logoff** to log off and exit the C-Hub Administration Console.

Starting the C-Hub

There are two ways to start a c-hub. If there is valid c-hub information in the repository and the startup class is registered in the WebLogic Server Administration Console, the c-hub will start up when you start the WebLogic Server.

In this case, the c-hub will already be running when you log on to the C-Hub Administration Console. (After you log on, you can view the name and URL for the running c-hub by clicking on **Configuration** in the left navigation panel, and then clicking on the **Hub** tab if it is not already displayed.)

If there is valid c-hub information in the repository but no active c-hub is registered with the server, the Hub configuration screen displays a Start HUB button. After you log on, you can click the **Start HUB** button to start the c-hub (based on the information stored in the Repository).

For more information on configuring and starting the C-Hub, see Chapter 2, "Setting Up the C-Hub," and *Setting Up the WebLogic Process Integrator Environment* in *BEA WebLogic Collaborate Getting Started*.

Stopping the C-Hub

To the stop the c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click **Shutdown HUB**.

Logging Off the C-Hub Administration Console

To log off the C-Hub Administration Console, click **Logoff** in the left navigation bar.

Overview of the C-Hub Administration Console

The C-Hub Administration Console allows you to monitor and configure the entities in a c-hub. The following sections provide a high-level view of the Administration Console configuration and monitoring features and summarize how the user interface works:

- A Quick Look at C-Hub Configuration with the Administration Console
- Relationships and Dependencies Among C-Hub Configuration Objects
- A Quick Look at C-Hub Monitoring

A Quick Look at C-Hub Configuration with the Administration Console

You can define and configure a c-hub and associated trading partners, documents, conversations, and c-spaces. In general, the task of configuring an item consists of the following steps:

1. Click **Configuration** in the left navigation bar.
2. Click the tab for the item you want to configure (c-hub, trading partners, documents, conversations, or c-spaces).

Note: You must first create a c-hub before you can do any other configuration tasks. For this release, WebLogic Collaborate supports only one c-hub per repository, so the Administration Console allows only one c-hub to be created.

3. Fill in the form fields (as described in the tables related to each screen on the Administration Console) and click **Add/Update** to add a new entity or update an existing one.

   Note that many of the configurable items rely on other items being configured appropriately, and you can navigate between screens to ensure all aspects of a c-hub are fully defined. For example, you can link to definitions for Trading Partners from the C-Spaces screen, or to definitions of Conversations from the Document Definitions screen.

In general, you must fill in all fields except those fields marked as “optional” in the field description tables in this Online Help.

Figure 8-3 Example of a Configuration Screen (Trading Partner Configuration)

For the majority of c-hub configuration screens, the following buttons are available:

- **Add/Update** adds information on the current form into the system if it does not exist or updates the information in the system if the information currently exists.
- **Remove** removes the information in the current form from the system. A dialog box will be displayed requesting confirmation. (See “Relationships and Dependencies Among C-Hub Configuration Objects” on page 8-9.)

- **Query** filters (searches) the information displayed. Currently only a simple query on the name is supported. (The query is case-sensitive.) The syntax for queries is:

  * — returns all

  name* — returns names that begins with name

  name — returns an exact match for name

- **Reset** resets the current form fields

- If there are more than 10 items (rows) available for viewing, **Next** and **Previous** links are displayed immediately beneath the table. No more than 10 items are shown per page. For example, on the Trading Partner tab shown in Figure 8-3, only four trading partners are defined on the c-hub (Partner1, Partner2, caller and callee) so no Next/Previous links are shown. If there were more than 10, the Next and Previous links would be displayed beneath the table of trading partners to let you navigate through large numbers of trading partners. The Next/Previous feature is available for all configurable c-hub entities (trading partners, document definitions, message definitions, logic plug-ins, and so on).

The rest of this Online Help provides detailed instructions on how to define and configure each of the c-hub items including the c-hub itself, trading partners, documents, c-spaces, and conversations.

Relationships and Dependencies Among C-Hub Configuration Objects

C-Hub configuration requires that various types of references be established among the objects, and that some objects act as containers for others. For example, a role in a conversation must *reference* a trading partner; whereas, a conversation definition *contains* roles. In the process of configuring a c-hub, you will quickly establish many dependencies among the objects. It is important to understand these dependencies and how various objects relate to one another.

This becomes particularly important if you want to *remove* an object. For example, you need to understand that if you remove a conversation, you are also removing all the roles you set up for that conversation.

For the current release, the Administration Console does not prevent you from removing objects with dependencies nor does it provide a detailed list of those dependencies for an object you are about to remove. The following table explains all references and containment relationships that can exist among c-hub objects. Use this table as a guide to track the dependencies among objects in your c-hubs.

Table 8-2 Relationships and Dependencies Among C-Hub Objects

C-Hub Object	References	Is Referenced By	Contains
Hub			All c-hub objects
Document Definition		Message Definitions	
Message Definition	Documents	Roles	
Logic Plug-in		Business Protocol Definitions	
Business Protocol Definition	Logic Plug-Ins		
Business Protocol	Business Protocol Definitions	C-Spaces	
Trading Partner			Trading Partner Protocols
Trading Partner Protocol	Business Protocol Definitions		
Conversation Definition			Roles
Role		Conversation Definitions	Message Definitions
C-Space			Business Protocol URLs Subscriptions

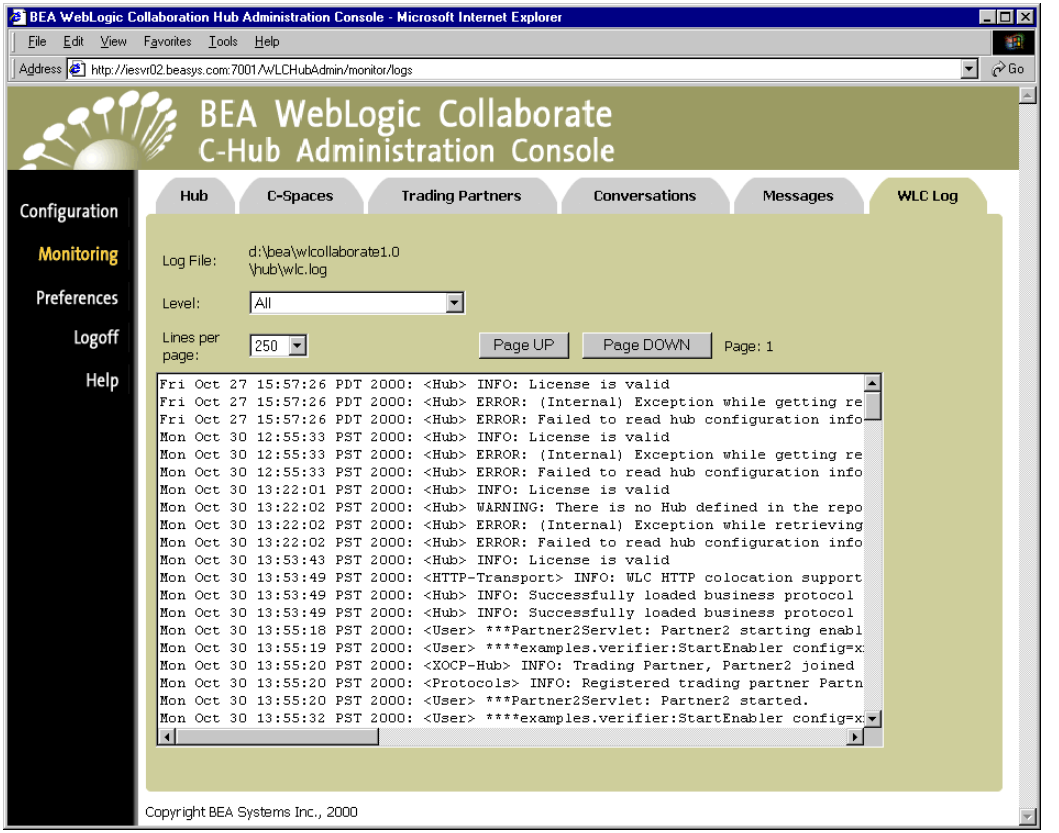
A Quick Look at C-Hub Monitoring

The C-Hub Administration Console provides a view into various reports and statistics you can use to help you track and monitor a running c-hub. In general, the task of monitoring an item consists of the following steps:

1. Click **Monitoring** in the left navigation bar.
2. Click the tab for the item or information you want to monitor (c-hub, c-spaces, trading partners, conversations, WLC log, or error logs).

   The appropriate information is displayed.

Figure 8-4 Example of a Monitoring Screen (Error Log)



Note that you can often access related items by navigating among the items on the screens in addition to using the tabs. For example, you can link to trading partners from the c-spaces screen, and so on.

For details on c-hub monitoring, see Chapter 18, “Monitoring the C-Hub.”

What’s Next?

The following sections describe in detail how to configure and monitor a c-hub:

- Chapter 9, “Creating and Modifying C-Hubs.”
- Chapter 10, “Configuring Document Definitions for a C-Hub.”
- Chapter 11, “Configuring Message Definitions for a C-Hub.”
- Chapter 12, “Working with Logic Plug-Ins.”
- Chapter 13, “Working with Business Protocol Definitions.”
- Chapter 14, “Working with Trading Partners.”
- Chapter 15, “Setting Up Conversations.”
- Chapter 16, “Working with C-Spaces.”
- Chapter 17, “Setting Preferences.”
- Chapter 18, “Monitoring the C-Hub.”

9 Creating and Modifying C-Hubs

The following sections provide key concepts and procedures for configuring and working with c-hubs on the C-Hub Administration Console:

- What Is a C-Hub?
- Creating a New C-Hub
- Modifying an Existing C-Hub
- Shutting Down a Running C-Hub
- Removing an Existing C-Hub
- Loading Data Into the C-Hub or Exporting C-Hub Data to a File
- Monitoring a C-Hub

What Is a C-Hub?

E-market is the generic term for an environment or container through which businesses can conduct B2B e-commerce. The c-hub, as the centerpiece of WebLogic Collaborate, provides an e-market environment. As such, the c-hub is responsible for routing messages between various WebLogic Collaborate c-enabler components and for managing the lifecycle of the conversations between the participating trading partners. The c-hub offers a set of services to conduct, manage, and orchestrate conversations (collaborations) between trading partners. These conversations are

defined by message vocabularies, business models, process classifications, roles, and other metadata. The WebLogic Collaborate e-market environment consists of one or more collaboration spaces, and a central c-hub.

A *collaboration space* (also known as a *c-space*) is an abstract entity in which trading partners conduct and coordinate conversations for a specific business purpose.

A *c-hub* is a point of control and exchange that represents an e-market owner. The c-hub enables an e-market owner to create and host a c-space. The e-market owner, through the c-hub, can host any number of c-spaces concurrently, with each c-space supporting any number of trading partners. Trading partners can participate in any number of conversations. Each trading partner uses a c-enabler to communicate with other trading partners, with all communication going through the c-hub.

Collaborators can communicate with each other by means of the c-hub. The c-hub provides shared services to BEA participant (c-enabler) nodes such as conversation and subscription management, security, administration, routing, local logging, and XML services.

For a complete explanation of the c-hub, trading partners, conversations, c-spaces, and other related concepts, see *BEA WebLogic Collaborate Getting Started*, particularly the Overview.

Creating a New C-Hub

You must first create a c-hub before you can do any other configuration tasks, such as configuring trading partners, documents, conversations and so on. There are two ways to create a new c-hub:

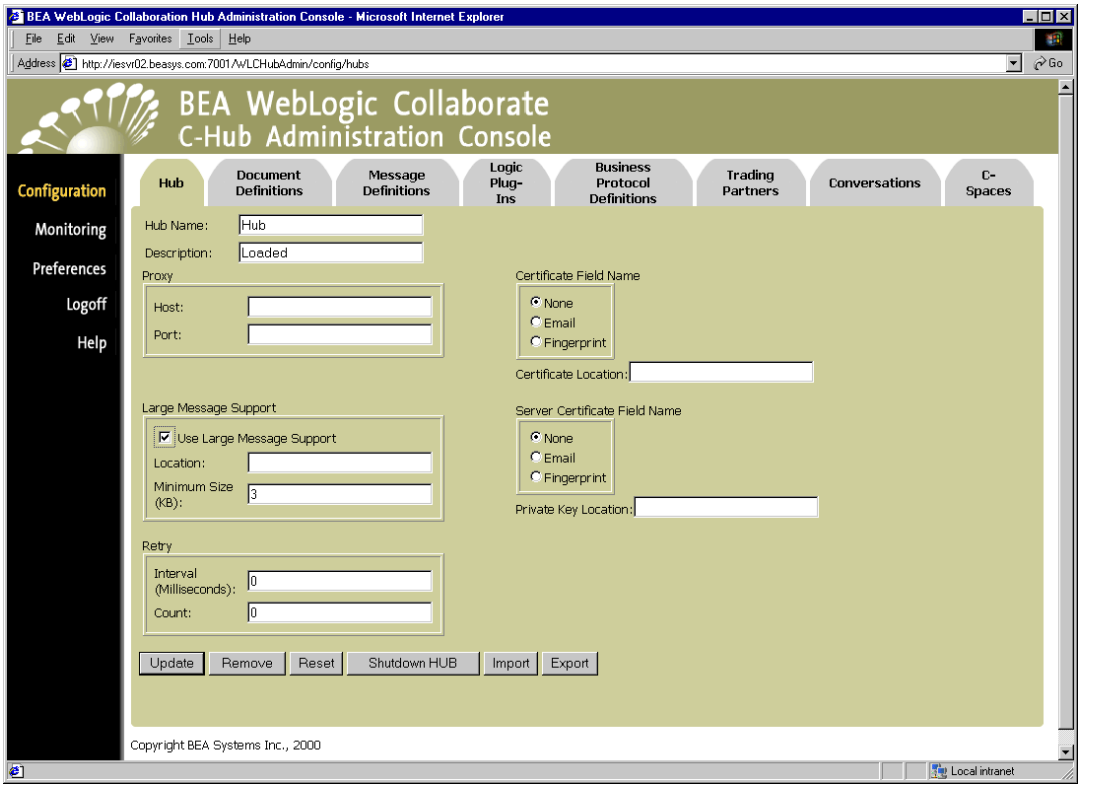
- You can create a new c-hub with the Administration Console user interface (See “Creating a New C-Hub Through the Administration Console” on page 9-3.)
- You can create a new c-hub by bulk-loading the data from a c-hub XML file via the Import/Export button on the Hub tab. (See “Creating a New C-Hub by Importing an XML File” on page 9-5.)

Creating a New C-Hub Through the Administration Console

To create a new c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Hub** tab to display the C-Hub configuration screen (if it is not already displayed).

Figure 9-1 C-Hub Configuration



3. Fill in the fields as described in the following table.

   Note: For this release, WebLogic Collaborate supports only one c-hub per repository, so the Administration Console allows only one c-hub to be created.)

Table 9-1 C-Hub Fields

Field	Description
Hub Name	Enter a name for the c-hub—this can be any meaningful name. (Limit is 256 characters.)
Description	Provide a brief description of the c-hub. (Limit is 256 characters.) (optional)
Proxy	■ **Host** - Enter the address of the proxy server used for the c-hub, if any. (For example, `myproxy.mycompany.com`.) ■ **Port** - Enter the port number for the proxy server.
Large Message Support	■ **Use Large Message Support** - Check this box if you want to use large message support. When this box is checkmarked, it indicates that large message support functionality is enabled. (By default, large message support is off.) ■ **Location** - If you are using large message support, indicate the directory for the large message support system working files here. ■ **Minimum Size** - If you are using large message support, indicate the minimum size for a message before the large message support functionality is enabled. (Minimum 3K)
Retry	■ **Interval** - Indicate the amount of time you want the c-hub to wait before trying to send the same message again if unsuccessful the previous time. The default is 1000 milliseconds. ■ **Count** - Indicate the number of times you want the c-hub to try to send a message.
Certificate Field Name	Select the certificate field you want to use for mapping trading partner certificates to WLS users. The default is *none*. If none is chosen, the WLC system defaults to “email” field.
Certificate Location	Enter the full path name of the server certificate field (optional, required only if SSL is used).

Table 9-1 C-Hub Fields

Field	Description
Server Certificate Field Name	Select the certificate field you want to use for authenticating remote SSL host. The default is *none*. If *none* is chosen, the WebLogic Collaborate system defaults to "email" field.
Private Key Location	Enter the full path name of the private key (optional, required only if SSL is used).

4. Click **Create** to create a new c-hub.

   You can also click **Reset** to discard your current changes and start again.

Creating a New C-Hub by Importing an XML File

As an alternative to creating a new c-hub through the user interface fields, you can *import* into the c-hub repository any properly formatted c-hub XML data file that describes the c-hub you want to create. C-hub XML data files for all WebLogic Collaborate examples are available for import. For more information on using both the import and export features, see "Loading Data Into the C-Hub or Exporting C-Hub Data to a File" on page 9-8.

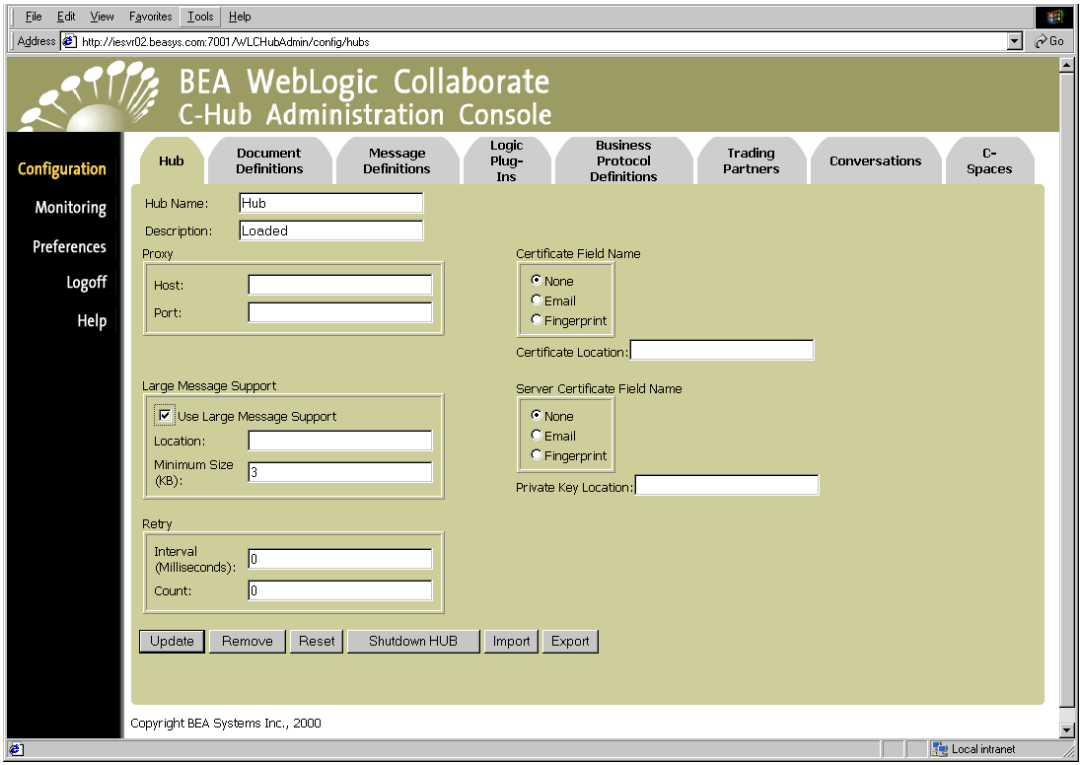
Modifying an Existing C-Hub

You can modify the definition for an existing c-hub. (Note that modifying the name of an existing c-hub is not allowed.)

To modify the definition for an existing c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Hub** tab to display the c-hub configuration screen (if it is not already displayed).

Figure 9-2 Modifying a C-Hub Configuration



3. Edit the description or URL as needed.
4. Click **Update** to update the c-hub.

   You can also click **Reset** to discard your current changes and start again.

Shutting Down a Running C-Hub

You can *shut down* or stop a c-hub that is currently running from either the c-hub Configuration tab or the c-hub Monitoring tab:

1. Click on **Monitoring** in the left navigation bar, then click on the **Hub** tab (if not already displayed).

   or

   Click on **Configuration** in the left navigation bar, then click on the **Hub** tab (if not already displayed).

2. Click the **Shutdown Hub** button.

   A confirmation dialog asks if you want to continue with the shutdown.

3. Click **OK** to confirm and shut down the c-hub, or click **Cancel** if you decide not to perform the shutdown.

Removing an Existing C-Hub

You can remove an existing c-hub. This operation can be performed on a running c-hub; however, we recommend that you shut down the c-hub first.

To remove an existing c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Hub** tab to display the c-hub configuration screen (if it is not already displayed).
3. Click **Remove**.

   A confirmation dialog is displayed. If you click **OK**, the current c-hub is removed from the database.

Note: When you remove a c-hub, you are removing all objects that it contains (document definitions, message definitions, logic plug-ins, trading partners, and so on.) If you want to save the current c-hub configuration data, be sure to export the data to a c-hub XML data file before you remove the c-hub. (See "Exporting Data for a C-Hub to an XML File" on page 9-12.) See also Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

Loading Data Into the C-Hub or Exporting C-Hub Data to a File

You can use the WebLogic Collaborate C-Hub Administration Console to import and export c-hub data. (The Administration Console automatically accesses the Bulk Loader to accomplish this.) As an alternative to creating a new c-hub through the user interface fields described in "Creating a New C-Hub Through the Administration Console" on page 9-3, you can *import* into the Administration Console any properly

formatted c-hub XML data file that describes the c-hub you want to create. You can also *export* data you have input through the Administration Console from the current c-hub repository to an XML file for later use.

A typical scenario is one in which you create a first c-hub by means of the Administration Console Hub tab, then export that new c-hub description to a c-hub XML data file. As you create new c-hubs through the Administration Console, you export them to c-hub XML data files for safekeeping. Once you have established a collection of c-hub XML data files, you can import them as needed.

You can also begin by importing a default c-hub XML data file to use as a template for creating new c-hubs.

You can also use the bulk loading utility by working directly with the Bulk Loader configuration files and DOS or UNIX commands on the command line.

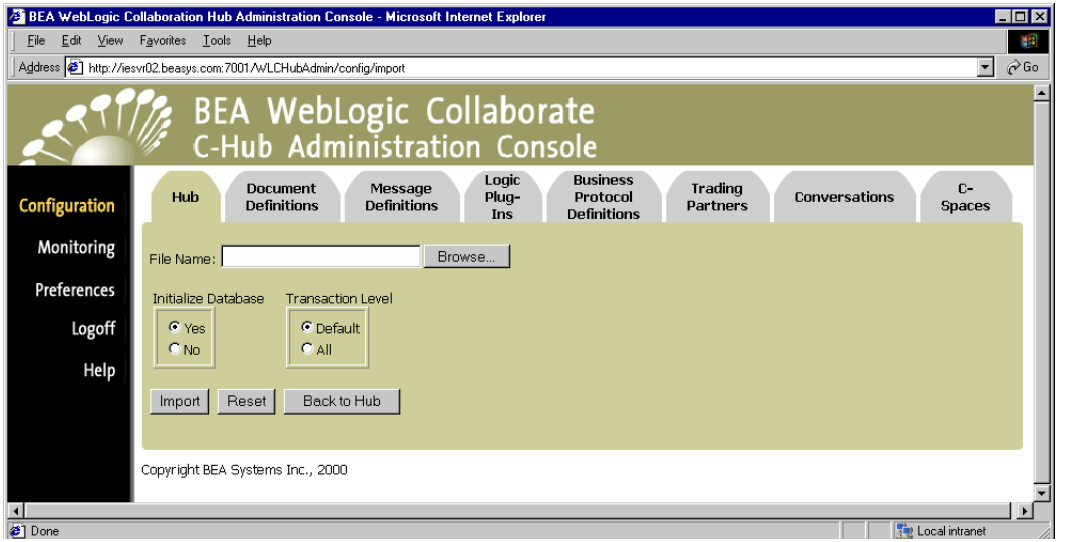
For more information on this, see Chapter 3, “Working with the Bulk Loader.”

Importing a C-Hub XML File

To import a c-hub XML file into the c-hub repository:

1. Make sure you are on the Hub configuration or monitoring tab.
2. Click **Import** to display the Import screen.

Figure 9-3 Import Screen



3. Indicate the full pathname of the c-hub XML file you want to import as described in the following table.

Table 9-2 C-Hub XML File Import Fields

Field	Description
Filename	Enter the full pathname of the c-hub XML file you want to import. You can type the pathname directly into the field or click **Browse** and use the file browser to navigate to the appropriate file.
Initialize Database	■ Select **Yes** to clear the existing c-hub data from the database during the import. In this case, the imported data will *replace* the existing data. ■ Select **No** to retain the existing c-hub data in the database during import. In this case, the imported data will be *added* to the existing data.
Transaction Level	Indicate the transaction level you want to use when importing this file. The transaction level you specify determines what actions to take if an error is detected during the import. The options are: ■ Select **All** to perform the import process as a single transaction. (We suggest using this option with small repositories.) If invalid data is detected in any of the c-hub entities as represented in the data file, the bulk loader utility rolls back the entire transaction and stops. The repository is left in exactly the same condition as it was before you started the import, with no data removed or added. ■ Select **Default** to perform the import with multiple transactions. (We suggest using this default setting for large repositories.) If you choose this option, the import initiates a transaction for each of the following entities: c-space, collaborator (a trading partner in a c-space), conversation definition, trading partner, document definition, transport protocol, module, business protocol definition, extended property set, message definition. If invalid data is detected during any one of these transactions, the bulk loader rolls back the current transaction only and continues processing for the next transaction.

4. Click **Import** to load the data from the c-hub XML file into the c-hub repository. This causes a new c-hub to be created from the c-hub XML file you imported. The data you import displays in the Administration Console.

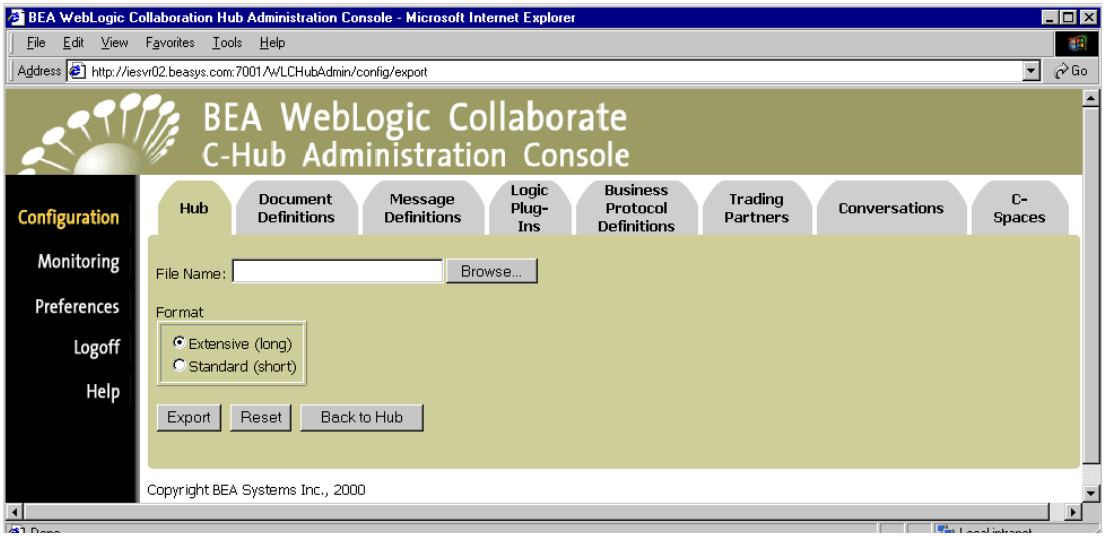
   (You can also click **Reset** to discard your current changes and start again.)

Exporting Data for a C-Hub to an XML File

To export data for an existing c-hub from a repository created with the Administration Console to an XML file:

1. Make sure you are on the Hub configuration or monitoring tab.
2. Click **Export** to display the Export screen.

Figure 9-4 Export Screen



3. Indicate the full pathname of the c-hub XML file you want to export as described in the following table.

Table 9-3 C-Hub XML File Export Fields

Field	Description
Filename	Enter the full pathname of the c-hub XML file you want to create or overwrite. You can type the pathname directly into the field or click **Browse** and use the file browser to navigate to an appropriate location and file.
Format	Indicate the *format* by clicking on either **Extensive** or **Standard**. (Use the Extensive format if you plan to migrate the data to another system or environment.)

4. Click **Export** to export the data from the C-Hub Administration Console into the c-hub XML file indicated. This causes the XML file to be created (or overwritten).

   (You can also click **Reset** to discard your current changes and start again.)

Monitoring a C-Hub

You can monitor the trading activity on a running c-hub by viewing reports on c-spaces, trading partners, conversations, statistics, and error logs. For information on how to monitor a c-hub, see Chapter 18, “Monitoring the C-Hub.”

CHAPTER

10 Configuring Document Definitions for a C-Hub

The following sections provide key concepts and procedures for configuring and working with document definitions with the C-Hub Administration Console:

- What Is a Document Definition?
- Adding New Document Definitions to the C-Hub
- Viewing Document Definitions on a C-Hub
- Modifying an Existing Document Definition
- How Do Document Definitions Relate to Message Definitions?
- Removing a Document Definition

What Is a Document Definition?

Trading partners in a c-space use a set of agreed-upon XML *document definitions* as a way of exchanging information. *Document definitions* are related to *roles* and *conversations*. That is, you can define entities that play particular roles (such as buyer or seller) that engage in an exchange of information called a *conversation*. Document definitions can be included in message definitions which are, in turn, assigned to roles in conversations. The information exchange (conversation) entails sending and receiving the agreed-upon message definitions (which include documents). A document definition is the blueprint or schema that defines a valid document.

This topic explains how to add new document definitions to a c-hub or modify the configurations for existing ones.

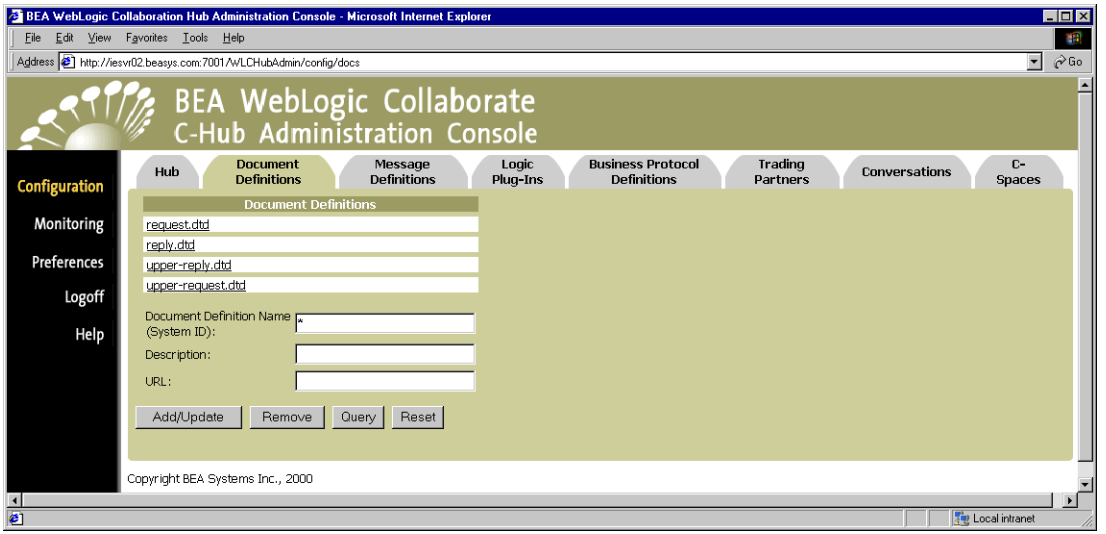
Adding New Document Definitions to the C-Hub

To add a new document to the c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Document Definitions** tab to display the Document Definitions configuration screen.

   In this screen you can enter document definition information that is used to create documents that trading partners will exchange in a conversation.

Figure 10-1 Document Definitions Configuration - Main Screen



From this screen, you can describe definitions of documents that are used inside conversations coordinated by the current c-hub. For this release, only XML

Document Type Definitions (DTD) are allowed to describe document definitions.

3. Fill in a Document Definition Name, description, and URL as described in the following table.

Table 10-1 Document Definitions Fields

Field	Description
Document Definition Name	The document type description (DTD) system identifier (Limit is 256 characters.)
Description	A description of the document (Limit is 256 characters.) (optional)
URL	URL to indicate where the document is located. (Limit is 80 characters.)

4. Click **Add/Update** to add a new document or update an existing one.

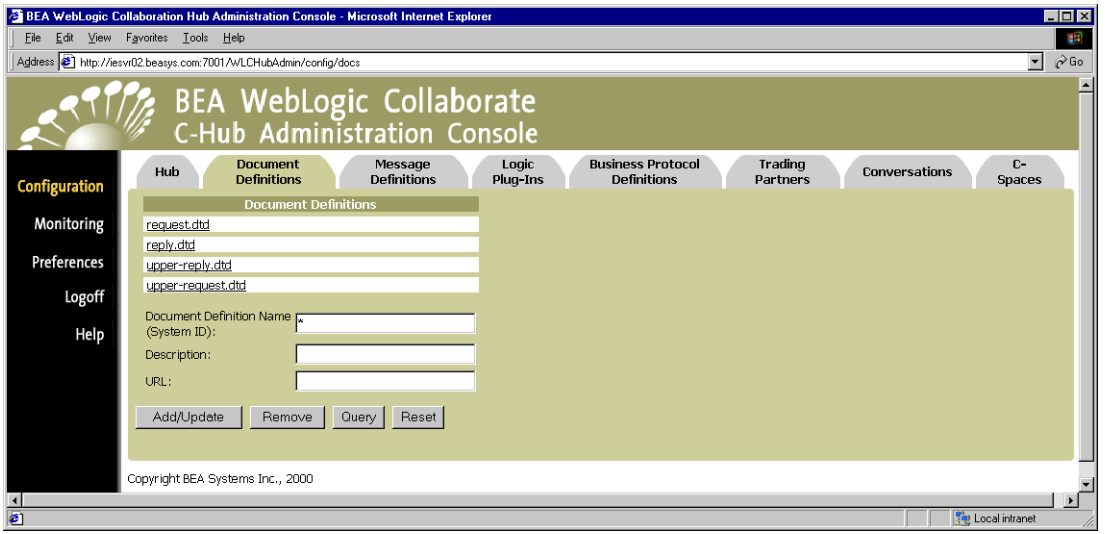
   (You can also click **Reset** to discard your current changes and start again.)

Viewing Document Definitions on a C-Hub

To view the document definitions on a c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Document Definitions** tab to display the Document Definitions configuration screen.

Figure 10-2 Document Definitions Configuration - Main Screen



The Document Definitions configuration screen shows the document definitions defined in the c-hub.

Using the Query Feature to Find Document Definitions

Notice the * in the Document Definition Name field. This is a special wildcard character used for queries. In the above screen, the table shows all the document definitions on the c-hub. You can type text strings in the Document Definition Name field and do queries on them to refine the table display. For instance, to display just

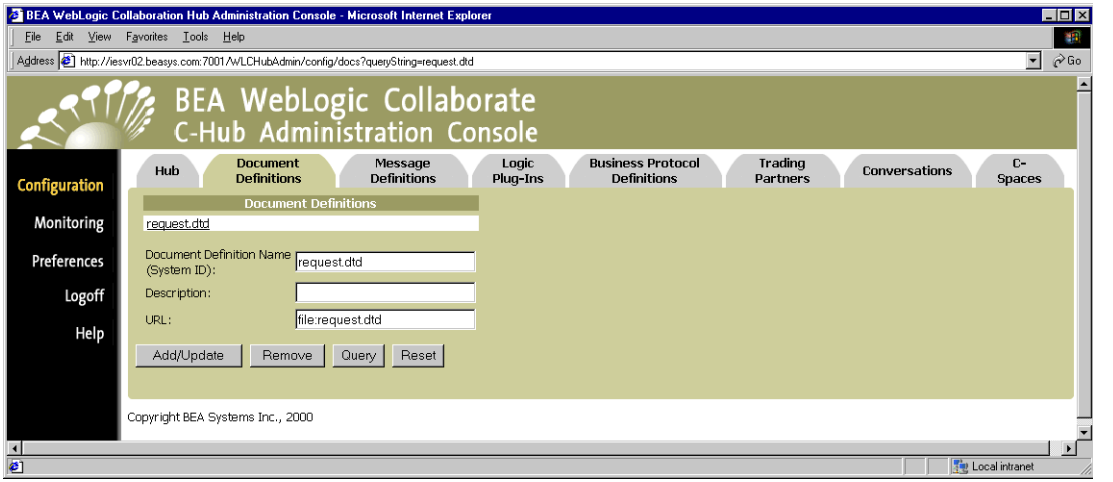
document definitions that start with “re”, you can input `re*` in the Document Definition Name field and click **Query**. (The query is case-sensitive.) To display information just for `request.dtd`, you can enter `request.dtd` in the Document Definition Name field and click **Query**, or just click on the `request.dtd` name in the *Document Definitions* list.

Modifying an Existing Document Definition

You can modify any or all of the settings for an existing document.

From the main Document Definitions configuration tab, click on a document definition name under *Document Definitions* to display the document definition for that document. (As an example, the following figure shows the document definition screen for a document definition named “request.dtd”.)

Figure 10-3 Document Definition for an Individual Document



You can modify the description or URL for a document definition from this screen.

Update the fields as needed and click **Add/Update** to save the changes.

How Do Document Definitions Relate to Message Definitions?

Document definitions are used in *message definitions*. For information on how to create a multi-part message definition using document definitions as message parts, see Chapter 11, "Configuring Message Definitions for a C-Hub."

Removing a Document Definition

To remove a document definition:

1. Click **Configuration** in the left navigation bar.
2. Click the **Document Definitions** tab to display the Document Definitions configuration screen.
3. In the Document Definition Name field, enter the full name of the document definition that you want to remove, and click **Remove**.

   or

   Click on the document definition you want to remove to display the configuration details for that document definition. On this screen, click **Remove**.

   A confirmation dialog is displayed. If you click **OK**, the specified document definition is removed and you are returned to the main Document Definitions configuration screen.

Note: If the object you remove *references* or *is referenced by* other objects (for example, if a message definition is using a document type you are about to remove), those references will be removed. If the object you remove *contains* other objects, those objects will be removed. See Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

CHAPTER

11 Configuring Message Definitions for a C-Hub

The following sections provide key concepts and procedures for configuring and working with message definitions on the C-Hub Administration Console:

- What Is a Message Definition?
- Adding New Message Definitions to the C-Hub
- Viewing Message Definitions on a C-Hub
- Modifying an Existing Message Definition
- How Do Message Definitions Relate to Roles and Conversations?
- Viewing the Role to Which a Message Definition Is Assigned
- Viewing the Conversation in Which a Message Definition Is Used
- Removing a Message Definition

What Is a Message Definition?

A business message is the basic unit of communication exchanged between trading partners in a conversation. The *message definition* defines the business content (business documents and attachments) of the business message. A message definition

consists of ordered message parts which can have a content type of *binary* or *XML*. An XML message part defines a business document and requires a document definition. A binary type message part defines an attachment and requires no other information.

This topic explains how to add new message definitions to a c-hub or modify the configurations for existing ones.

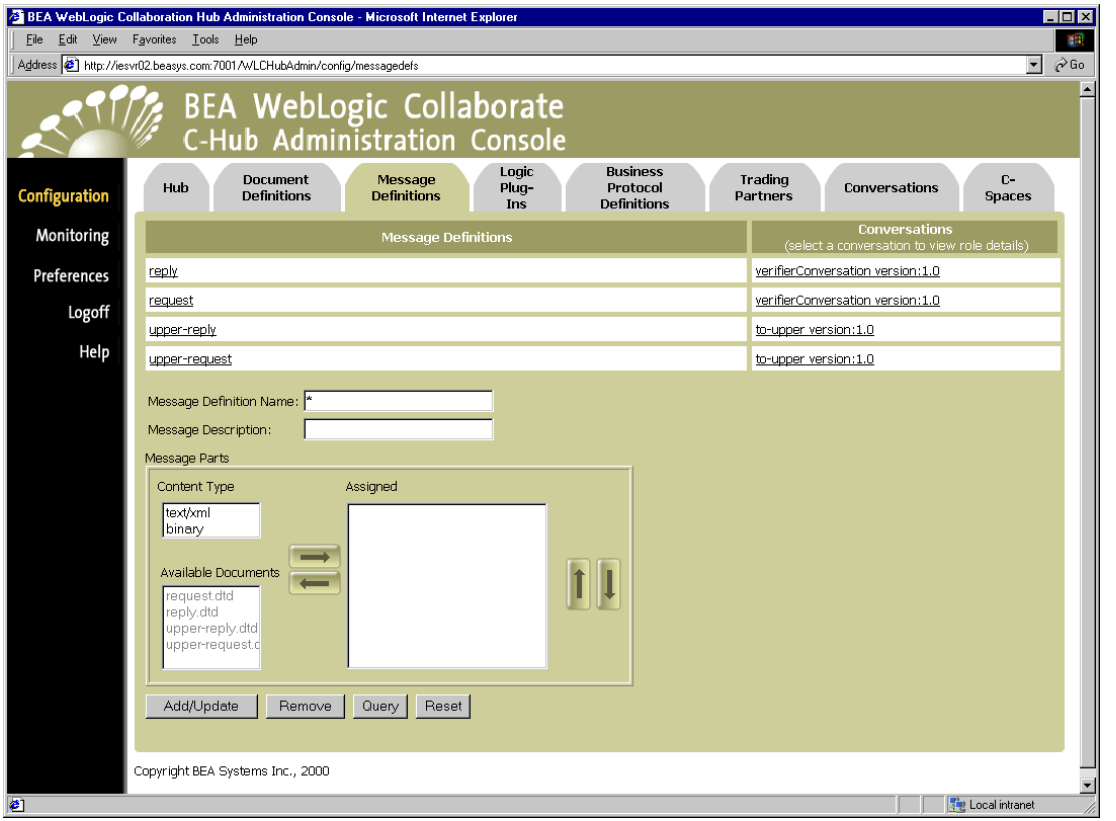
Adding New Message Definitions to the C-Hub

To add a new message definition to the c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Message Definitions** tab to display the Message Definitions configuration screen.

This screen is used to enter message definition information.

Figure 11-1 Message Definitions Configuration - Main Screen



From this screen, you can create message definitions of messages that are used inside conversations coordinated by the current c-hub.

3. Fill in the fields as described in the following table.

Table 11-1 Message Definition Fields

Field	Description
Message Definition Name	Enter a name for the message definition. Limit is 256 characters.
Message Description	Enter a description of the message definition. Limit is 256 characters. (optional)
Message Parts	You can build a multi-part message by using the right and left arrows to move both binary message definitions and XML documents into the *Assigned* Message Parts. Create each message part as follows: ■ **Content Type** - Select either `text/xml` or `binary` as the content type. ■ **Documents** - If content type is `text/xml`, select a document (`*.dtd`) and use the right arrow key to move it into the *Assigned* list for Message Parts. **Note:** Message definitions use document definitions as message parts. For information on how to create document definitions, see Chapter 10, "Configuring Document Definitions for a C-Hub." ■ **Message Parts** - This list can contain XML documents and any binary message definitions you want to include in the message definition. Use the up and down arrows to put the message parts into the appropriate order or sequence. Use the right and left arrow buttons to move documents and binary definitions in and out of the Message Parts list.

4. Click **Add/Update** to add a new message definition.

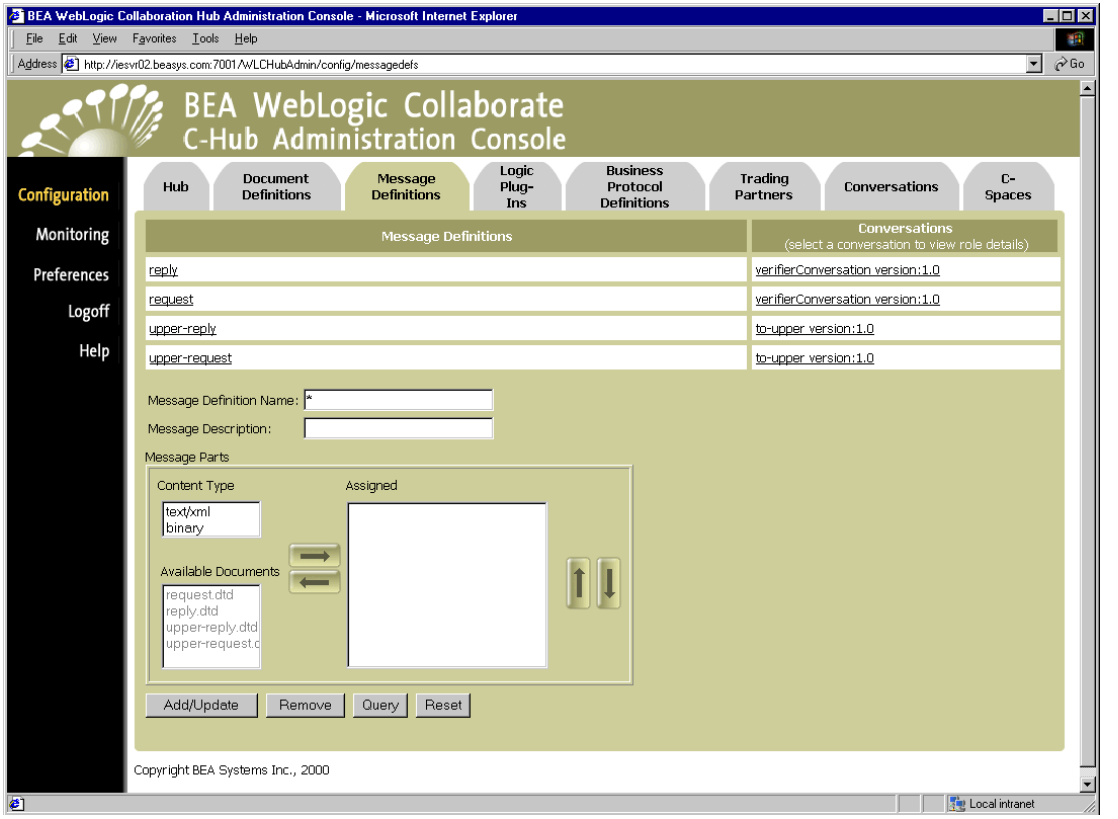
   (You can also click **Reset** to discard your current changes and start again.)

Viewing Message Definitions on a C-Hub

To view the message definitions on a c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Message Definitions** tab to display the Message Definitions configuration screen.

Figure 11-2 Message Definitions Configuration - Main Screen



The Message Definitions configuration screen shows the messages that are defined in the c-hub and conversations that use them.

Using the Query Feature to Find Message Definitions

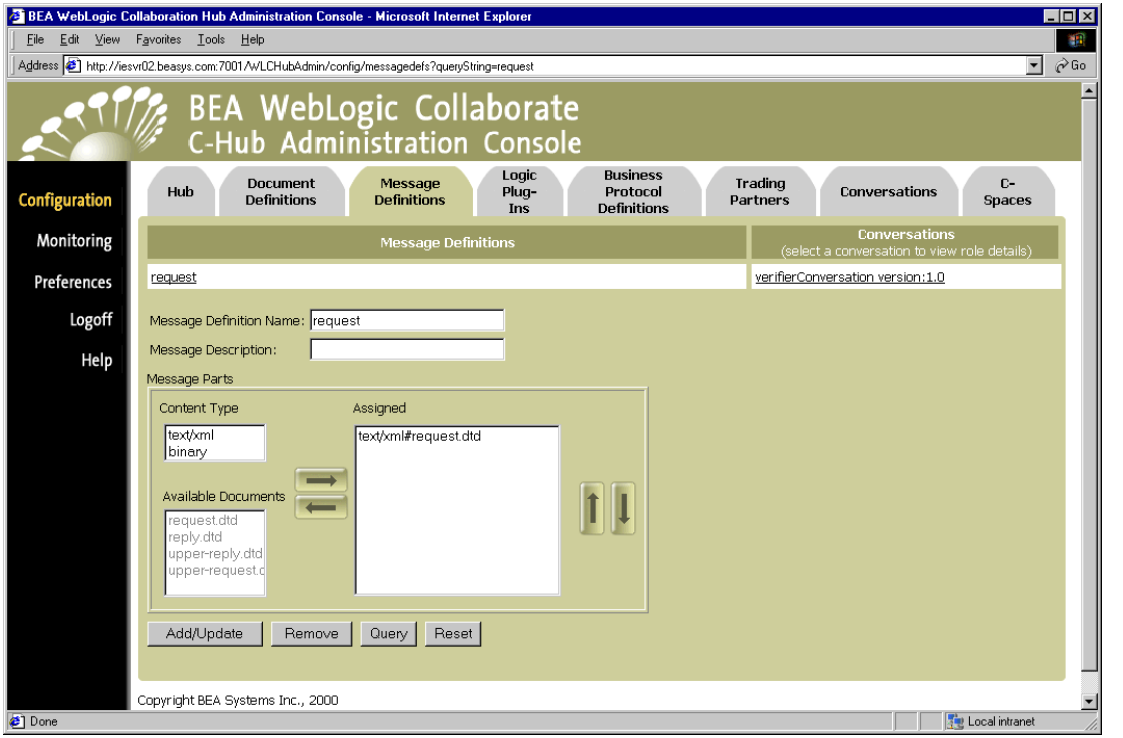
Notice the * in the Message Definition Name field. This is a special wildcard character used for queries. In the above screen, the table shows all the message definitions on the c-hub. You can type text strings in the Message Definition Name field and do queries on them to refine the table display. For instance, to display just message definitions that start with “re”, you can input `re*` in the Message Definition Name field and click **Query**. (The query is case-sensitive.) To display information just for `request`, you can enter `request` in the Message Definition Name field and click **Query**, or just click on the `request` name in the *Message Definitions* list.

Modifying an Existing Message Definition

You can modify any or all of the settings for an existing message definition.

From the main Message Definitions configuration tab, click on a message definition name under *Message Definitions* to display the definition for that message. (As an example, the following figure shows the message definition screen for a message named “request”.)

Figure 11-3 Definition for an Individual Message



From this screen, you can modify the message definition, message description, content types for document definitions, and which document definitions to include in the message definition.

To modify a message definition, update the fields as needed and click **Add/Update** to save the changes.

How Do Message Definitions Relate to Document Definitions?

Message definitions use document definitions as message parts for business messages. For information on how to create document definitions, see Chapter 10, "Configuring Document Definitions for a C-Hub."

How Do Message Definitions Relate to Roles and Conversations?

Message definitions are used in *roles* which are, in turn, used in *conversations*. When you set up a conversation, you need to define roles for it (such as "requestor" and "replier"), and assign message definitions to those roles. For example, a "requestor" might be set up to send a `request` message and receive a `reply` message. Whereas, the "replier" might be set up to receive a `request` message and send a `reply` message.

For information on how to define and configure a conversation and its participant roles, see Chapter 15, "Setting Up Conversations."

Viewing the Role to Which a Message Definition Is Assigned

To view the role to which a message definition is assigned, start from the main Message Definitions configuration tab and click on the name of a conversation in the *Conversations* list. This takes you to the configuration information for that particular conversation (on the Conversations tab), where you can view details about the conversation and roles associated with it.

Note: If a message definition has not yet been assigned to a conversation, the *Conversations* column will show [NONE] indicating no conversations for that document.

Viewing the Conversation in Which a Message Definition Is Used

To view the conversations in which a message definition is used, start from the main Message Definitions configuration tab and click on the name of a conversation in the *Conversations* list. This takes you to the configuration information for that particular conversation (on the Conversations tab).

For information on how to define and configure a conversation, see Chapter 15, “Setting Up Conversations.”

Removing a Message Definition

To remove a message definition:

1. Click **Configuration** in the left navigation bar.
2. Click the **Message Definitions** tab to display the Message Definitions configuration screen.
3. In the Message Definition Name field, enter the full name of the message definition that you want to remove, and click **Remove**.

   or

   Click on the message definition you want to remove to display the configuration details for that message definition. On this screen, click **Remove**.

   A confirmation dialog is displayed. If you click **OK**, the specified message definition is removed and you are returned to the main Message Definitions configuration screen.

Note: If the object you remove *references* or *is referenced by* other objects (for example, if a conversation is using a message definition you are about to remove), those references will be removed. If the object you remove *contains* other objects, those objects will be removed. See Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

CHAPTER

12 Working with Logic Plug-Ins

The following sections provide key concepts and procedures for configuring and working with logic plug-ins on the C-Hub Administration Console:

- What Are Logic Plug-Ins?
- Setting Up a Logic Plug-In
- Viewing Logic Plug-Ins on a C-Hub
- Modifying an Existing Logic Plug-In
- Removing a Logic Plug-In

For more information on logic plug-ins, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*.

What Are Logic Plug-Ins?

The message flow through the c-hub begins with an *incoming* message from a c-enabler, proceeds to the routing service, and ends with the c-hub sending an *outgoing* message. This flow is usually described from the perspective of the c-enabler client, rather than the c-hub. In this context, the incoming message flow through the router is considered to be part of *send-side* routing (because the message is sent by the c-enabler to the c-hub). The outgoing message sent by the c-hub is considered to be available for *receive-side* filtering (because it is received by the c-enabler from the c-hub).

WebLogic Collaborate provides a router and a filter for each business protocol that it supports. Each router and each filter consists of a chain of logic plug-ins. You can add your own logic plug-ins to the routers and filters. The plug-in code must conform to BEA guidelines and interfaces. These *logic plug-ins* (components in filters and routers) allow you as a c-hub owner to provide additional processing (*rules*) for the information passing through the c-hub.

- On the send-side, the *router* determines the trading partner(s) to whom a message should be routed. Router plug-ins, either provided by BEA or written by the c-hub owner, can add or remove recipients from the list.
- On the receive-side, the *filter* receives a message destined for a single trading partner recipient and decides whether to send the message. C-hub owners can add their own filter plug-ins, or use those provided by BEA.

However, you are not limited to using the plug-ins strictly for filtering and routing tasks. As a c-hub owner, you can customize the "filter" and "router" plug-ins for many purposes other than filtering and routing, such as sampling, validating, auditing, billing, and so on. The following table shows some simple examples of the kinds of additional processing rules you might want to implement and perform on data flowing through the c-hub.

Table 12-1 Examples of Using Router and Filter Logic Plug-Ins on the C-Hub for Various Types of Additional Processing

Plug-In to Use	Example Rule to be Implemented
Router	"If a computer chip order over $1M is submitted, make sure that NewChipCo is one of the recipients."
Filter or Router	"After January 1, 2000, no orders should be sent to OldChipCo."
Filter or Router	"Log all senders/recipients of messages for billing purposes."
Filter or Router	"Sample 1 out of every N messages of type X for standards compliance."
Filter or Router	"For messages of type X, how many are conversation version 1 versus conversation version 2?"

For more information on what logic plug-ins are and how to construct and use them, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*.

Setting Up a Logic Plug-In

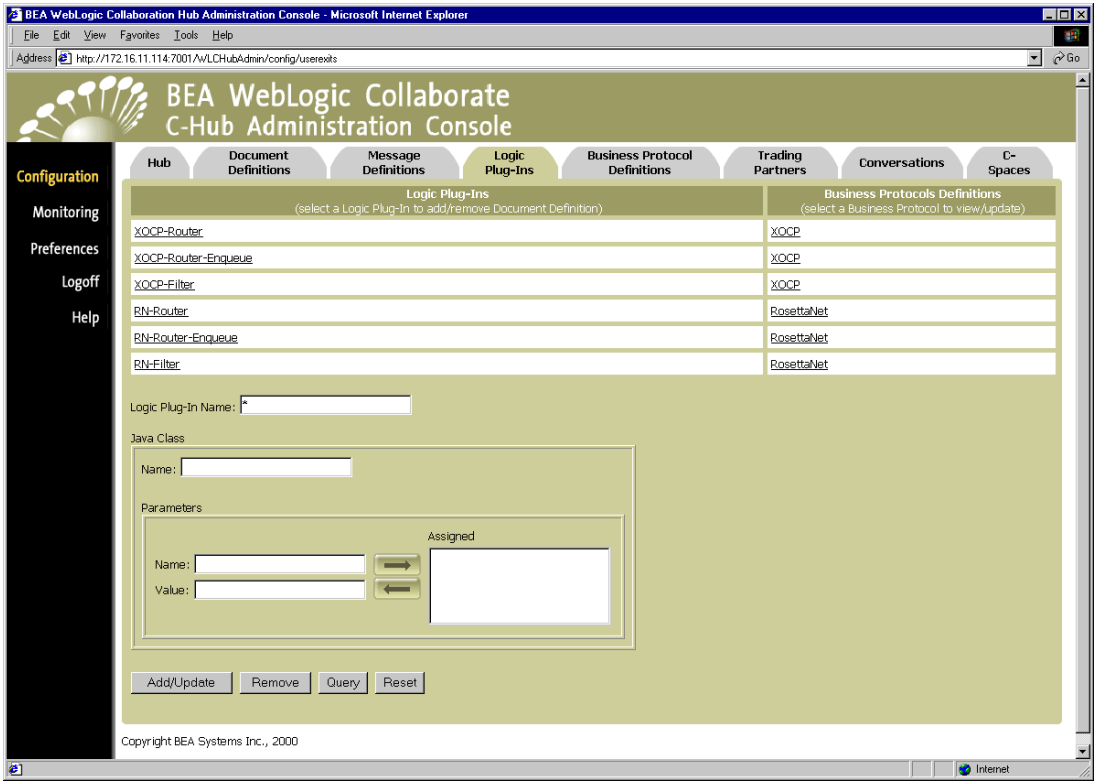
Setting up a new logic plug-in definition on the c-hub consists of the following tasks:

- Step 1. Name Plug-In and Provide Parameters.
- Step 2. Assign the Logic Plug-In to a Business Protocol Definition.

Step 1. Name Plug-In and Provide Parameters.

1. Click **Configuration** in the left navigation bar.
2. Click the **Logic Plug-Ins** tab to display the Logic Plug-Ins configuration screen.

Figure 12-1 Logic Plug-Ins Configuration



The Logic Plug-Ins list shows the logic plug-ins currently defined and available for use. The Business Protocols Definitions list shows the protocol assigned for each plug-in.

Note: If a logic plug-in has a parameter name of `bea.hidden` and a value of `true`, then it will not show up on the Logic Plug-Ins screen list unless you have checked *Show hidden logic plug-ins* in the Preferences screen. (For

more information on the "hidden logic plug-ins" setting, see Chapter 17, "Setting Preferences.")

3. Fill in the fields as described in the following table.

Table 12-2 Logic Plug-Ins Fields

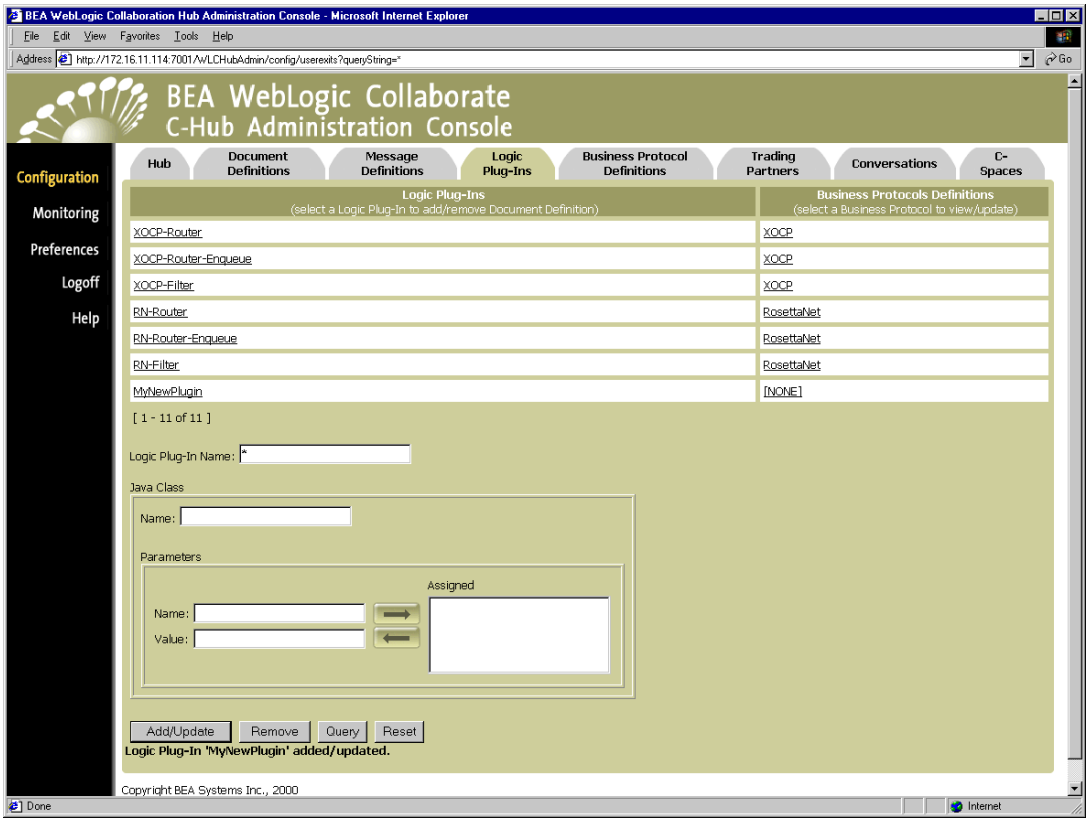
Field	Description
Logic Plug-In Name	Enter the name of a new logic plug-in you want to create. The logic plug-in is one that you define based on code you have written and what you want to track or accomplish with regard to the data passing through your c-hub.
Java Class	■ **Name -** Enter the name of the Java class specifying your implementation of this particular logic plug-in. ■ **Parameters -** In the **Parameter Name** field, enter the name of a parameter to be read at initialization. For example, you might want to specify the name of a database where you want to store data, the type of message you want to audit, or a time to start or stop operations. In the **Parameter Value** field, enter a value for your initialization parameter. When you have the parameter name and value specified as desired, use the right arrow button to move the parameter you have defined into the *Assigned* list box for Java class parameters.

4. Click **Add/Update** to create the new logic plug-in. Your new logic plug-in should show up in the *Logic Plug-Ins* list.

   (You can also click **Reset** to discard your current changes and start again.)

   Initially, your new logic plug-in will not be assigned to a business protocol, so [NONE] will show up under *Business Protocols* list for that logic plug-in.

Figure 12-2 New Logic Plug-In Added



Step 2. Assign the Logic Plug-In to a Business Protocol Definition.

You can create new "flavors" of an existing business protocol based on the particulars of your enterprise, but this is not a task that can be handled in the C-Hub Administration Console.

You will probably be accessing the Business Protocol Definitions tab in the C-Hub Administration Console primarily for the purpose of assigning logic plug-ins (components in filters and routers) to an existing protocol like XOCP.

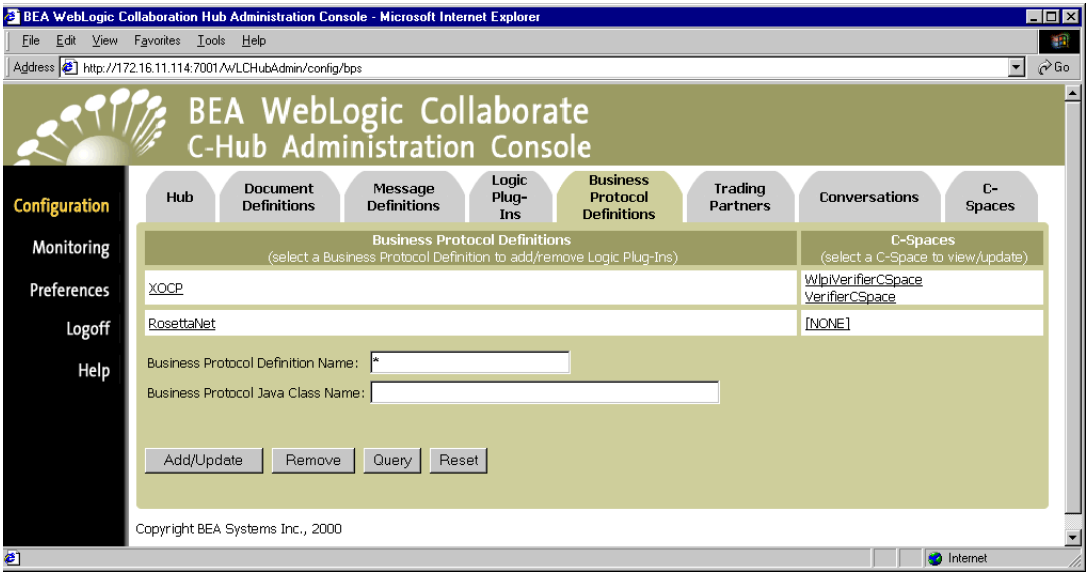
For information on developing logic plug-ins to use in routers and filters, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*. For more information on using XOCP filters and routers, see Chapter 7, "Routing and Filtering XOCP Business Messages."

To assign a logic plug-in to a business protocol:

1. Click on the business protocol name under the *Business Protocols Definitions* list on the Logic Plug-Ins tab. In our example, we select XOCP. (If you haven't assigned a business protocol yet, you can click on [NONE] which shows up under *Business Protocols* list for that logic plug-in.)

This brings up the main Business Protocol Definitions Screen.

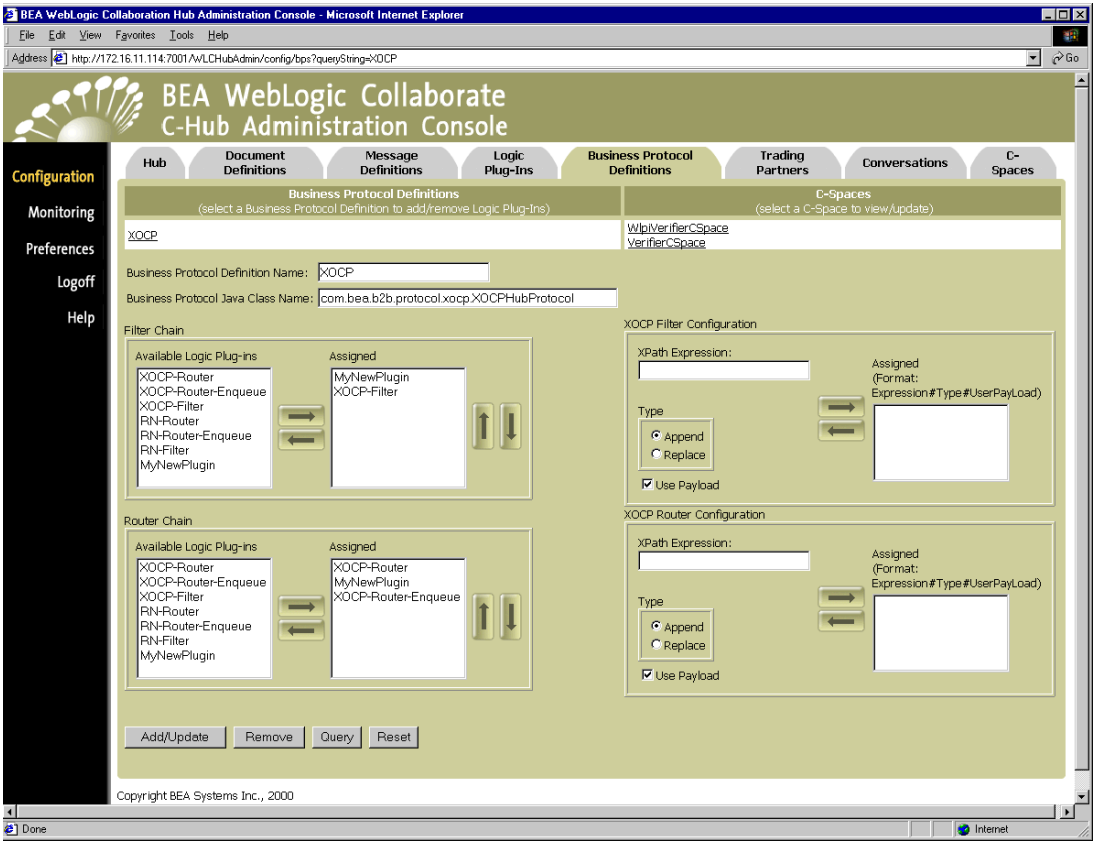
Figure 12-3 Business Protocol Definitions Screen



2. Click on the business protocol you want to use. The C-Hub Administration Console displays the Business Protocol Definitions screen where you can assign logic plug-ins (components in filters and routers) to the selected business protocol. (For our “MyNewPlugin” example, we choose XOCP.) Logic plug-ins you have defined on the Logic Plug-in tab show up on the detail screen for a

business protocol definition as available logic plug-ins in the Filter Chain and Router Chain group areas. You can move them into the *Assigned* list box to assign them to filter and router chains.

Figure 12-4 Assign Logic Plug-Ins to a Business Protocol Definition



Note: If a logic plug-in has a parameter name of `bea.hidden` and a value of `true`, then it will not show up on Available Logic Plug-ins lists in the Filter and Router Chain lists unless you have checked *Show hidden logic plug-ins* on the Preferences screen.

3. To create Filter Chains and Router Chains, use the arrow buttons to move the logic plug-ins you want to assign to this business protocol into the *Assigned* lists. Once you have the appropriate filters and routers showing in the "Assigned" list box, use the up and down arrow buttons to order the chain of filters and routers appropriately.

   Note: XOCP filters and routers can be used only for XOCP business protocol definitions. Likewise, RosettaNet filters and routers can be used only for RosettaNet business protocol definitions.

4. To define an XPath expression for the XOCP filter logic plug-in or the XOCP router logic plug-in, use the XOCP Filter Configuration and XOCP Router Configuration groups, respectively, as described in the following table.

   Note: If you define an XPath expression in the XOCP Filter Configuration group, this XPath expression will be added to the XOCP filter logic plug-in. Similarly, if you define an XPath expression in the XOCP Router

Configuration group, this XPath expression will be added to the XOCP router logic plug-in.

Table 12-3 XPath Fields for XOCP Filter and XOCP Router Configurations

Field	Description
XPath Expression	Enter an XPath expression in this text field. (XPath is the XML path language. An XPath expression is a string that specifies intended recipients of a business message using XPath syntax.) These XPath expressions are evaluated in sequence at run time against the message-context XML document, which has the content of the message along with meta information about the message, such as the sending and receiving trading partners, conversation information, and versioning. In the XOCP router logic plug-in, each XPath router expression can examine different parts of the message-context document and select a different set of recipient trading partners. The trading partners produced by each expression can either replace the previously generated set of recipient trading partners or add to the current set. In the XOCP filter logic plug-in, each XPath filter expression can examine different parts of the message-context document to determine whether or not to forward the message to the recipient trading partner. Each XPath filter expression can return `true` or `false` using different selection criteria. After an XPath filter expression returns `false`, the message is blocked from further evaluation and is not sent to the intended recipient. For example, you might want to filter messages for trading partners that are shippers so that they receive only shipping requests, while all other types of trading partners receive all messages: `(/c-hub/message/@message-def-name='Shipping Request' AND /c-hub/trading-partner/extended-property-set/business='shipper') OR (/c-hub/trading-partner/extended-property-set/business!='shipper')`
Type	Select either **Append** or **Replace**. If you choose Append, the results of evaluating the new XPath expression are added to the results of the evaluations of any preceding XPath expressions. If you choose Replace, the results of evaluating the new XPath expression replace the results of the evaluations of all preceding XPath expressions.
Use Payload	Select whether to use payload. (A checkmark indicates payload will be used.) If you select Use Payload, the generated message-context document will include message parts from the payload.

When you have defined the XPath expression you want to use and set the type and payload filtering options, click on the right arrow button to move the XPath

expression into the *Assigned* list. (You can use the left arrow button to remove an expression from the *Assigned* list.)

These fields and options work the same way for both XOCP Filter Configuration and XOCP Router Configuration.

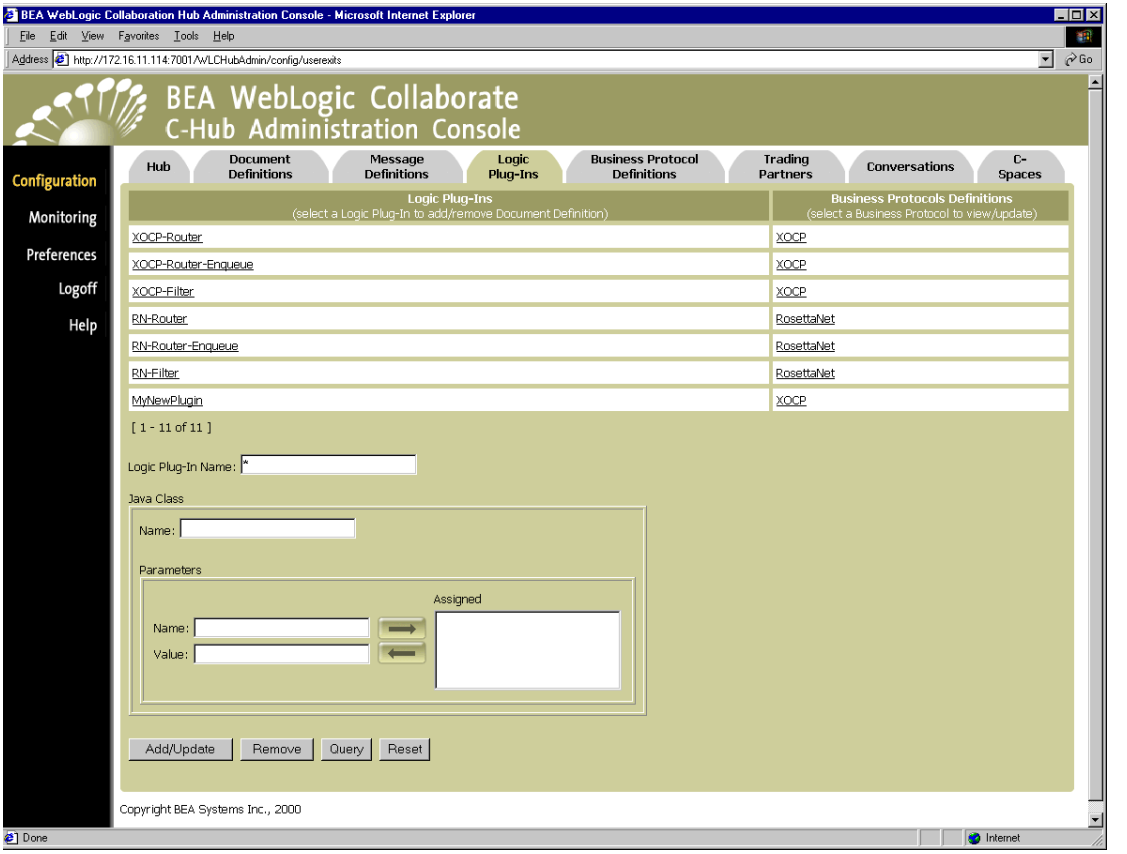
Note: When you use the Business Protocol Definitions tab to create an XPath expression, you are defining a *c-hub XPath expression.* C-hub XPath expressions are applied to all XOCP business messages flowing through the c-hub. (In contrast, you can also define XPath expressions for a particular trading partner which are applied only to XOCP business messages sent or received by a particular trading partner.)

5. When you have the appropriate logic plug-ins listed in the *Filters* and *Routers* lists, click **Add/Update** to assign these plug-ins to the selected business protocol.

   The plug-in assignments are processed, and you are returned to the main Business Protocol Definitions screen. A status message is displayed indicating that the business protocol assignment has been made to the appropriate logic plug-in.

If you now click the Logic Plug-Ins tab to display the main Logic Plug-Ins configuration screen, you should see your new plug-in assigned to the appropriate business protocol.

Figure 12-5 New Logic Plug-In Added to Plug-Ins List with Protocol Assignment



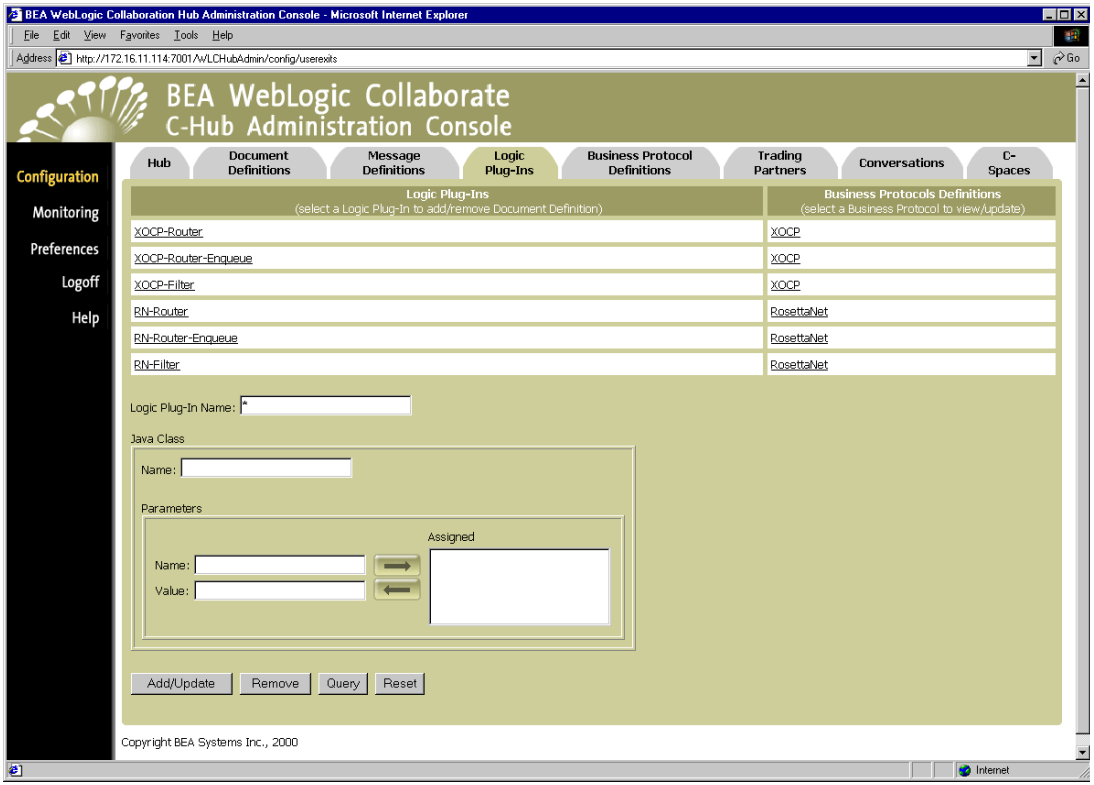
For more information on business protocols, see Chapter 13, “Working with Business Protocol Definitions.”

Viewing Logic Plug-Ins on a C-Hub

To view logic plug-ins on a c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Logic Plug-Ins** tab to display the logic plug-ins configuration screen.

Figure 12-6 Viewing Logic Plug-Ins



The Logic Plug-Ins configuration screen shows a table of logic plug-ins currently defined in the c-hub (except those configured as `bea.hidden`) and the business protocols associated with each plug-in.

Using the Query Feature to Find Logic Plug-Ins

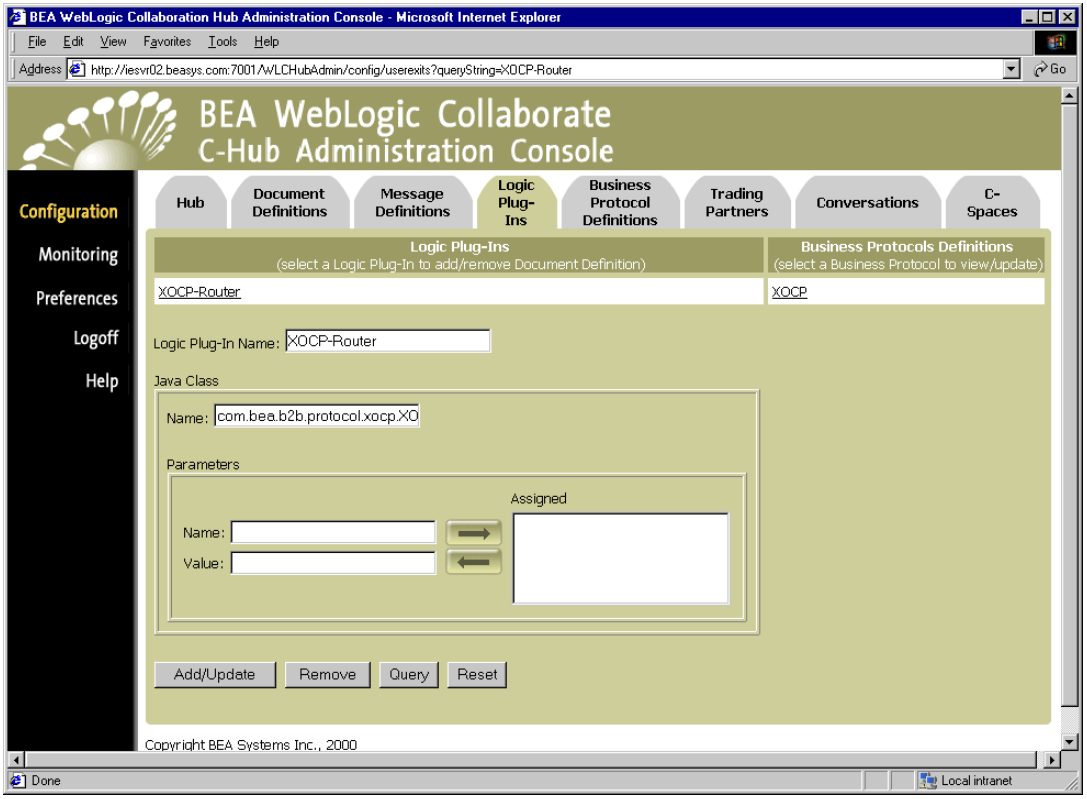
Notice the * in the Logic Plug-In Name field. This is a special wildcard character used for queries. In the above screen, the table shows all the logic plug-ins on the c-hub (except those configured as `bea.hidden`). You can type text strings in the Logic Plug-In Name field and do queries on them to refine the table display. For instance, to display just logic plug-ins that start with "XOCP", you can input `XOCP*` in the Logic Plug-In Name field and click **Query**. (The query is case-sensitive.) To display information just for `XOCP-Router`, you can enter `XOCP-Router` in the Logic Plug-In Name field and click **Query**, or just click on the `XOCP-Router` name in the *Logic Plug-Ins* list.

Modifying an Existing Logic Plug-In

You can modify any or all of the existing logic plug-ins set up on a c-hub.

From the main Logic Plug-Ins configuration tab, click on a logic plug-in name under *Logic Plug-Ins* to display the configuration for that plug-in. (As an example, the following figure shows the configuration screen for a logic plug-in named "XOCP-Router".)

Figure 12-7 Configuration for an Individual Logic Plug-In



You can modify the configuration for the selected plug-in from this screen. Simply redefine the name, parameters, or other attributes as needed, then click **Add/Update** to save the changes. (For details on the fields, see Table 12-2.)

Removing a Logic Plug-In

Warning: Do not remove the BEA-provided logic plug-ins for XOCP or RosettaNet. Removal of these plug-ins will disable the WebLogic Collaborate c-hub. You can add and remove your own, customized logic plug-ins as needed.

To remove a customized logic plug-in:

1. Click **Configuration** in the left navigation bar.
2. Click the **Logic Plug-Ins** tab to display the Logic Plug-Ins configuration screen.
3. In the Logic Plug-In Name field, enter the full name of the logic plug-in that you want to remove, and click **Remove**.

   or

   Click on the logic plug-in you want to remove to display the configuration details for that logic plug-in. On this screen, click **Remove**.

   A confirmation dialog is displayed. Click **OK** to remove the specified logic plug-in in. You are returned to the main Logic Plug-Ins configuration screen.

Note: If the object you remove *references* or *is referenced by* other objects (for example, if a business protocol is using a logic plug-in you are about to remove), those references will be removed. If the object you remove *contains* other objects, those objects will be removed. See Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

CHAPTER

13 Working with Business Protocol Definitions

The following sections provide key concepts and procedures for configuring and working with business protocol definitions on the C-Hub Administration Console:

- What Are Business Protocols?
- Assigning Logic Plug-Ins to Business Protocols Definitions
- Using the Query Feature to Find Business Protocol Definitions
- Adding a Business Protocol Definition
- Removing a Business Protocol Definition

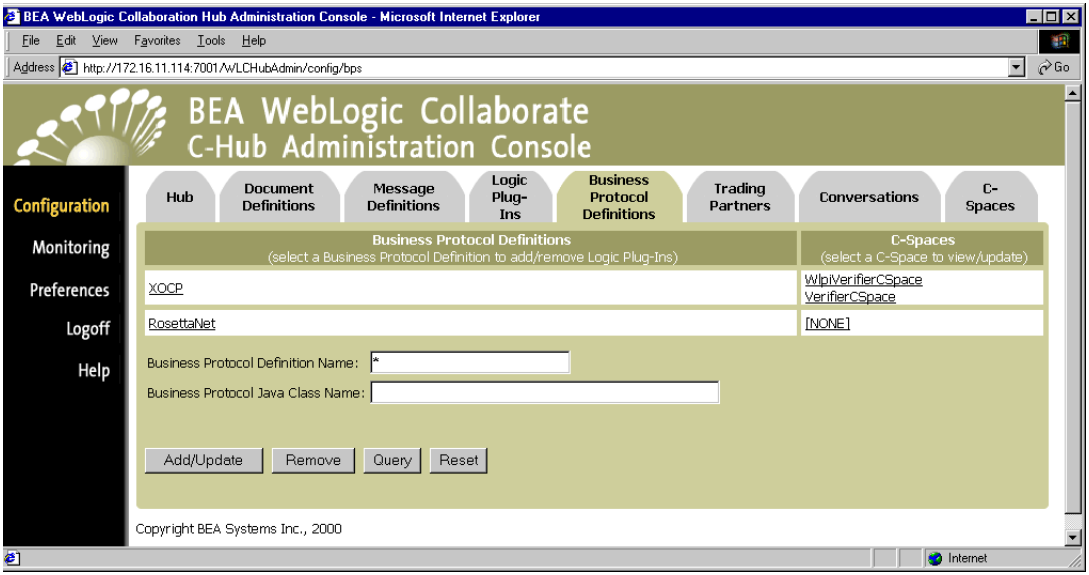
For more information on business protocol definitions, see the *BEA WebLogic Collaborate Developer Guide*.

What Are Business Protocols?

A business protocol defines how a business message will be processed by the c-hub, including how it is read off the wire; how it is routed to one or more recipients; and how it will be put back on the wire and sent to its destination. A business protocol also defines the details of persistence, retries, and quality of service, in general.

WebLogic Collaborate provides XOCP and RosettaNet protocol support. You can customize and extend these protocols beyond the out-of-the-box functionality they offer by adding your own logic plug-ins. (See Chapter 12, “Working with Logic Plug-Ins.”)

Figure 13-1 Business Protocol Definitions Screen



For more information on business protocols, see Chapter 6, “Configuring Business Protocols,” and Overview in *BEA WebLogic Collaborate Getting Started*.

Assigning Logic Plug-Ins to Business Protocols Definitions

You can create new “flavors” of an existing business protocol like XOCP or RosettaNet based on the particulars of your enterprise, but this is not a task that can be handled in the C-Hub Administration Console.

You will probably be accessing the Business Protocol Definitions tab in the C-Hub Administration Console primarily for the purpose of assigning logic plug-ins (components in filters and routers) to an existing protocol like XOCP.

For a detailed description on how to assign logic plug-ins to business protocols, refer to "Step 2. Assign the Logic Plug-In to a Business Protocol Definition." on page 12-7 in Chapter 12, "Working with Logic Plug-Ins."

For information on developing logic plug-ins to use in routers and filters, see Developing Logic Plug-Ins in the *BEA WebLogic Collaborate Developer Guide*. For more information on using XOCP filters and routers, see Chapter 7, "Routing and Filtering XOCP Business Messages."

Using the Query Feature to Find Business Protocol Definitions

Notice the * in the Business Protocol Definition Name field. This is a special wildcard character used for queries. In the above screen, the table shows all the business protocol definitions on the c-hub (except those configured as `bea.hidden`). You can type text strings in the Business Protocol Definition Name field and do queries on them to refine the table display. For instance, to display just business protocol definitions that start with "X", you can input `X*` in the Business Protocol Definition Name field and click **Query**. (The query is case-sensitive.) To display information just for `XOCP` protocols, you can enter `XOCP` in the Business Protocol Definition Name field and click **Query**, or just click on the `XOCP` name in the *Business Protocol Definitions* list.

Adding a Business Protocol Definition

To add a business protocol definition, you must use the Bulk Loader to load an appropriately defined XML configuration file.

For more information, see Chapter 3, "Working with the Bulk Loader."

Removing a Business Protocol Definition

Warning: Do not remove the BEA-provided business protocol definitions for XOCP or RosettaNet. Removal of these protocols will disable the WebLogic Collaborate c-hub. You can add and remove your own, customized business protocol definitions as needed.

To remove a customized business protocol definition:

1. Click **Configuration** in the left navigation bar.
2. Click the **Business Protocol Definitions** tab to display the Business Protocol Definitions configuration screen.
3. In the Business Protocol Name field, enter the full name of the business protocol definition that you want to remove, and click **Remove**.

   or

   Click on the business protocol definition you want to remove to display the configuration details for that business protocol definition. On this screen, click **Remove**.

   A confirmation dialog is displayed. If you click **OK**, the specified business protocol definition is removed and you are returned to the main Business Protocol Definitions configuration screen.

Note: If the object you remove *references* or *is referenced by* other objects (for example, if a logic plug-in is assigned to a business protocol definition you are about to remove), those references will be removed. If the object you remove *contains* other objects, those objects will be removed. See Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

CHAPTER

14 Working with Trading Partners

The following sections provide key concepts and procedures for configuring and working with trading partners on the C-Hub Administration Console:

- What Are Trading Partners?
- Creating a Trading Partner
- Defining XOCP Filters and Routers for a Trading Partner
- Defining Server-Side Security Values for a Trading Partner
- Assigning a Trading Partner to a TP Protocol
- Adding Extended Properties for a Trading Partner
- Viewing Trading Partners on a C-Hub
- Modifying an Existing Trading Partner
- Viewing Configuration Details for Trading Partners
- How Do Trading Partners Relate to C-Spaces?
- Removing a Trading Partner

For introductory information about trading partners, see Overview in *BEA WebLogic Collaborate Getting Started*. For more information about developing trading partner applications, see the *BEA WebLogic Collaborate Developer Guide*.

What Are Trading Partners?

A trading partner represents an entity, such as a company, that is authorized to participate in collaboration activity through BEA WebLogic Collaborate. A trading partner is authorized to participate in one or more predefined collaboration spaces (c-spaces). Within a c-space, a trading partner can participate in one or more conversations by being subscribed (authorized) to a role in one or more conversation definitions. Trading partners are registered on the c-hub level, so the c-hub administrator must first define and configure them, specifying such attributes as name, address, e-mail, phone, and fax information for each trading partner.

For a complete explanation of what trading partners are and how they relate to e-markets and collaboration spaces, see *BEA WebLogic Collaborate Getting Started*, particularly the Overview.

Creating a Trading Partner

To create a new trading partner:

1. Click **Configuration** in the left navigation bar.
2. Click the **Trading Partners** tab to display the Trading Partners configuration screen.
3. Fill in the fields as described in the following table.

Table 14-1 Trading Partners Fields

Field	Description
Trading Partner Name	Enter the name of a new trading partner you want to create or modify. (Limit is 256 characters.) Note that the trading partner name you enter in this field must be the same as the <trading-partner name> defined in the XML file for the c-enabler.
Description	Enter a brief description of the trading partner. (Limit is 256 characters.) (optional)

Table 14-1 Trading Partners Fields

Field	Description
Address	Enter the physical address of the trading partner. (Limit is 256 characters.) (optional)
Certificate Field Value	This is the certificate field value for the certificate-field name specified in the Hub tab. The value is used when *fingerprint* is chosen in the Hub tab. When *email* or *none* are chosen, then the email address is used. The value is used while mapping the trading partner certificate to a WebLogic Server (WLS) user. (This is a client-side security setting related to connections flowing from the trading partners' c-enablers to the c-hub. For information on how to set server-side security for connections flowing out from the c-hub, see "Defining Server-Side Security Values for a Trading Partner" on page 14-8.)
WLS User Name	WLS user field lists the WebLogic Server user the trading partner certificate should be mapped to while executing messages sent by that trading partner. (This is a client-side security setting related to connections flowing from the trading partners' c-enablers to the c-hub. For information on how to set server-side security for connections flowing out from the c-hub, see "Defining Server-Side Security Values for a Trading Partner" on page 14-8.)
Email	Enter the e-mail address for the trading partner. (Limit is 256 characters.)
Phone	Enter the phone number for the trading partner. (Limit is 256 characters.) (optional)
Fax	Enter the fax number for the trading partner. (Limit is 256 characters.) (optional)
State	Select Enabled or Disabled.

Figure 14-1 Trading Partners Configuration

4. Click **Add/Update** to create the new trading partner. Your new trading partner should show up in the *Trading Partners* list.

   (You can also click **Reset** to discard your current changes and start again.)

When you create a new trading partner, initially it will not be part of a collaboration space (c-space) until you assign it to one. This is indicated by the label [NONE] showing next to the newly added trading partner under the *C-Spaces* list. You can add trading partners to c-spaces that you set up on the C-Spaces tab. From the Trading Partners tab, you can click on the c-space for a trading partner (or [NONE] if none are

defined) and go directly to the C-Spaces tab. (Information on how to create c-spaces and add trading partners to them is described in the topic "Working with C-Spaces" on page 16-1.) You can skip to this task now, or continue working through the Trading Partner tasks in sequential order, depending on which entities you would like to set up first. When you later define c-spaces and add trading partners to them, those relationships will be reflected on the Trading Partners screen shown in Figure 14-1.

Similarly, when first created, the trading partner will not be assigned to a trading partner (TP) protocol, and the TP Protocol for the new trading partner will be [NONE]. (For more information on this, see "Assigning a Trading Partner to a TP Protocol" on page 14-10.)

Defining XOCP Filters and Routers for a Trading Partner

You can use XOCP filters and XOCP routers to control the flow of XOCP business messages exchanged among trading partners in a c-space. (Note that XOCP filters and XOCP routers can be used only for XOCP business protocol definitions.)

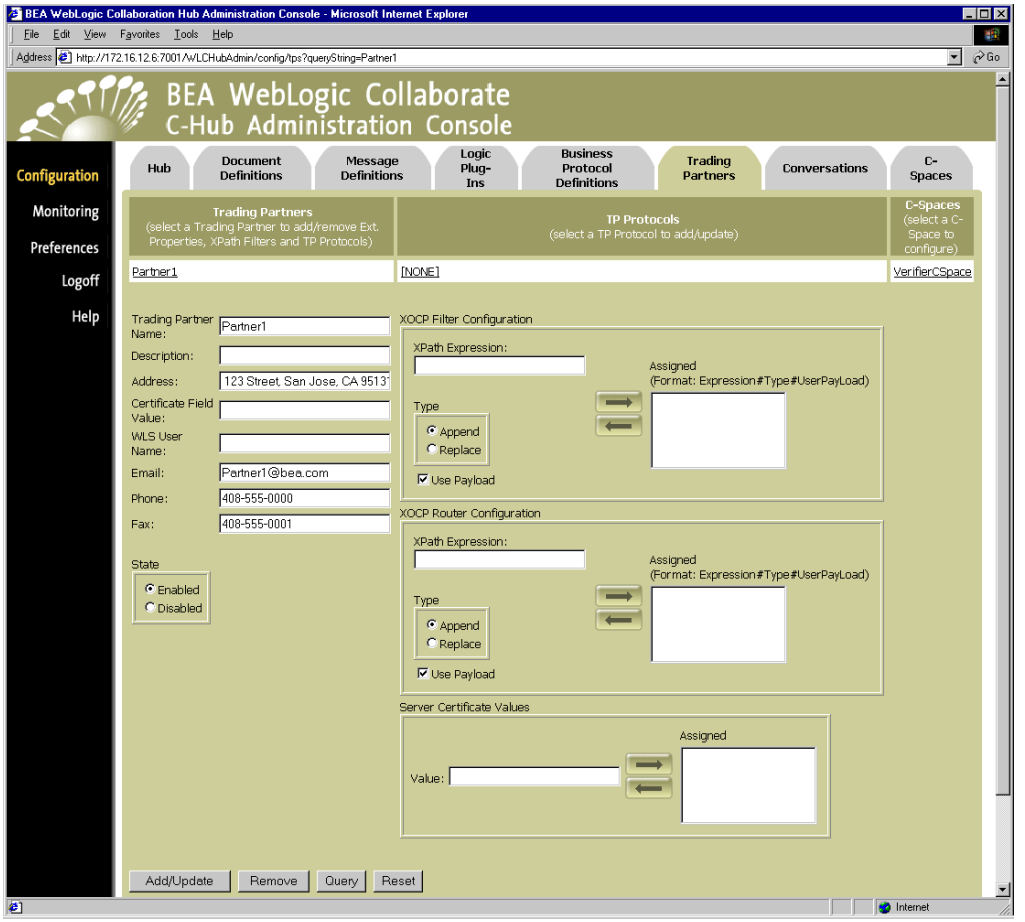
For more information about XOCP filters and XOCP routers, see Chapter 7, "Routing and Filtering XOCP Business Messages."

To define filters and routers for a trading partner:

1. From the main Trading Partner tab, click the trading partner to which you want to assign filters and routers.

   This brings up a screen showing the configuration details for the selected trading partner.

Figure 14-2 Details on a Trading Partner



2. From this screen you can configure XOCP filters and routers for a trading partner by defining XPath expressions as described in the following table.

   Note: If you define an XPath expression in the XOCP Filter Configuration group, this XPath expression will be added to the XOCP filter logic plug-in. Similarly, if you define an XPath expression in the XOCP Router Configuration group, this XPath expression will be added to the XOCP router logic plug-in.

Table 14-2 XPath Fields for XOCP Filter and XOCP Router Configurations

Field	Description
XPath Expression	Enter an XPath expression in this text field. (XPath is the XML path language. An XPath expression is a string that specifies intended recipients of a business message using XPath syntax.) These XPath expressions are evaluated in sequence at run time against the message-context XML document, which has the content of the message along with meta information about the message, such as the sending and receiving trading partners, conversation information, and versioning. In the XOCP router logic plug-in, each XPath router expression can examine different parts of the message-context document and select a different set of recipient trading partners. The trading partners produced by each expression can either replace the previously generated set of recipient trading partners or add to the current set. In the XOCP filter logic plug-in, each XPath filter expression can examine different parts of the message-context document to determine whether or not to forward the message to the recipient trading partner. Each XPath filter expression can return `true` or `false` using different selection criteria. After an XPath filter expression returns `false`, the message is blocked from further evaluation and is not sent to the intended recipient. For example, you might want to filter messages for trading partners that are shippers so that they receive only shipping requests, while all other types of trading partners receive all messages: `(/c-hub/message/@message-def-name='Shipping Request' AND /c-hub/trading-partner/extended-property-set/business='shipper') OR (/c-hub/trading-partner/extended-property-set/business!='shipper')`
Type	Select either **Append** or **Replace**. If you choose Append, the results of evaluating the new XPath expression are added to the results of the evaluations of any preceding XPath expressions. If you choose Replace, the results of evaluating the new XPath expression replace the results of the evaluations of all preceding XPath expressions.

Table 14-2 XPath Fields for XOCP Filter and XOCP Router Configurations

Field	Description
Use Payload	Select whether to use payload. (A checkmark indicates payload will be used.) If you select Use Payload, the generated message-context document will include message parts from the payload.

When you have defined the XPath expression you want to use and set the type and payload filtering options, click on the right arrow button to move the XPath expression into the *Assigned* list. (You can use the left arrow button to remove an expression from the *Assigned* list.)

These fields and options work the same way for both XOCP Filter Configuration and XOCP Router Configuration.

Note: When you use the detail screen for a trading partner to create an XPath expression, you are defining a *trading partner XPath expression*. Trading partner XPath expressions are applied only to XOCP business messages that the selected trading partner sends or receives. (In contrast, you can also define XPath expressions for your business protocol definition which are applied to all XOCP business messages flowing through the c-hub.)

3. When you have the XOCP filters and routers you want to use listed in the appropriate Chain lists, click **Add/Update** to update the trading partner to use the assigned XPath expressions.

   (You can also click **Reset** to discard your current changes and start again.)

Defining Server-Side Security Values for a Trading Partner

You can define *client-side* security values (certificate field value and WLS user name) from the main Trading Partners tab when you add a new trading partner (as described in “Creating a Trading Partner” on page 14-2.) Client-side security values relate to connections flowing from the c-enablers (trading partners) to c-hub.

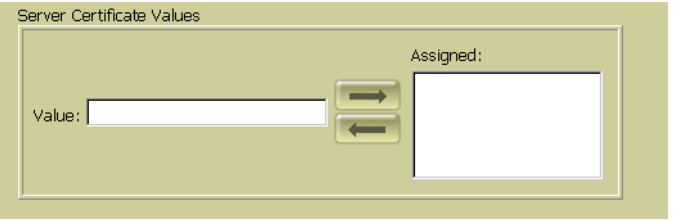
In addition to these client-side settings, you can also assign *server-side* security values to a trading partner. Server-side security values relate to connections flowing from the c-hub to the c-enablers (trading partners). This option is located on the screen that shows configuration details for a selected trading partner (along with the XPath expressions details).

To define Server-Side Security Values:

1. From the main Trading Partners tab, click the trading partner to which you want to assign filters and routers.

   This brings up a screen showing the configuration details for the selected trading partner. This screen includes the settings for server-side security.

Figure 14-3 Server-Side Security Option on Detailed Configuration Screen for a Trading Partner



2. Fill in the server certificate value as described in the following table.

Table 14-3 Trading Partners Security Fields

Field	Description
Server Certificate Value	Enter a server certificate value. The server certificate field value is used to authenticate message recipients to the SSL certificate holder obtained from the remote SSL host. The server certificate value is used while the c-hub is sending a message to a trading partner's c-enabler.
Assigned	When you have filled in the server certificate value, use the right arrow button to move it into the *Assigned* list. (You can use the left arrow to remove a value from the *Assigned* list.)

3. When you have the appropriate server certificate value in the *Assigned* list, click **Add/Update** to update the trading partner with the new value.

   (You can also click **Reset** to discard your current changes and start again.)

Assigning a Trading Partner to a TP Protocol

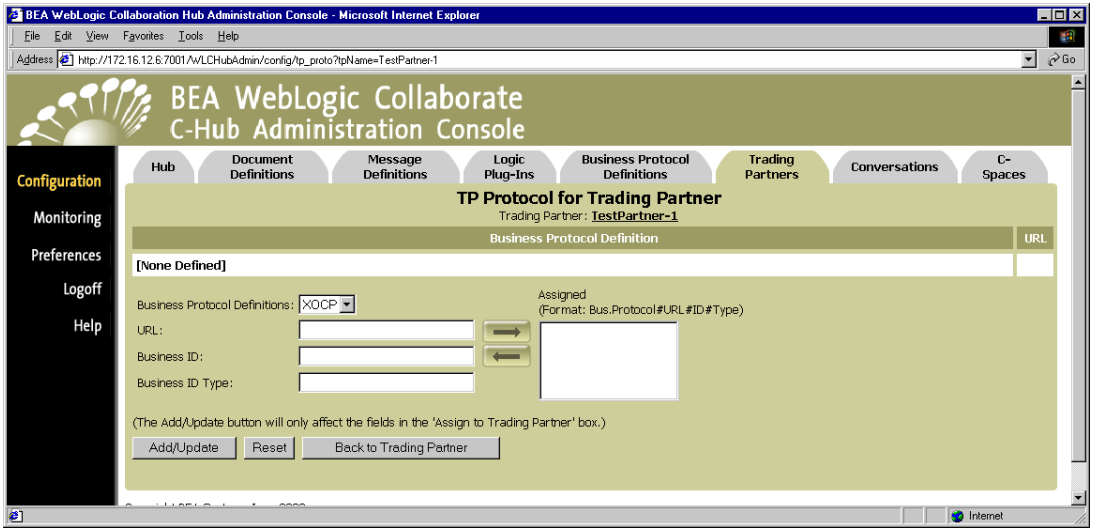
Trading Partner Protocol defines additional trading partner information needed for non-XOCP business protocol support.

To assign a trading partner to a trading partner (TP) protocol or modify an existing assignment:

1. From the main Trading Partners tab, click the TP protocol for the trading partner you want to modify. If this trading partner is not yet assigned to a TP protocol, click the [NONE] in the TP protocol list for the trading partner you want to assign.

This brings up the TP protocol assignment screen for the selected trading partner. (In our example, we have brought up the TP assignment screen for a trading partner called “TestPartner-1” as shown on the top of the screen.)

Figure 14-4 TP Protocol Assignment



(You can also use the Add/Update TP Protocol button on the configuration details screen for a trading partner to get to this “TP Protocol” screen.)

2. Fill in the fields as described in the following table.

Table 14-4 TP Protocol Assignment Fields

Field	Description
Business Protocol Definitions	Use the drop down menu to select the name of a business protocol definition with which to associate this partner.
URL	An external URL to where business messages should be routed. Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol. **Warning:** A URL for a c-space/business protocol combination can be used only by the c-hub and c-enablers. If customer-supplied software uses one of these URLs, messages will not be processed correctly.
Business ID	A business identifier to be used by an external trading partner, which is distinct from the internal WLC trading partner name.
Business ID Type	The type of the business identifier. For example, for RosettaNet, a "DUNS" number is used. The type names are under the control of the c-hub administrator and need to be synchronized with the related business protocol plug-in.

3. Click **Add/Update** to assign the trading partner to the TP protocol with the specified data.

   (You can also click **Reset** to discard your current changes and start again.)

Adding Extended Properties for a Trading Partner

The “Extended Properties for a Trading Partner” screen lets you add additional information that you want to associate with a trading partner. This additional data associated with a trading partner is expressed in the repository as XML sub-trees. The sub-tree root element is `extended-property-set`.

In the following example XML document generated from the repository, everything contained within the `extended-property-set` element is user-defined extended property information for a trading partner named "B2Binc."

```
<c-hub context="message-router">
…
<trading-partner name="B2Binc"
     email="sales@b2b.com"
     phone="408-111-222">
     <address>123 Main St., San Jose, CA 95131</address>
     <extended-property-set name="B2Binc">
          <business-contact>Joe Jackson</business-contact>
          <phone type="work">408-999-1212</phone>
          <phone type="cell">408-999-1234</phone>
          <city>San Jose</city>
          <state>California</state>
     </extended-property-set>
</trading-partner>
…
</c-hub>
```

To add extended properties to a trading partner:

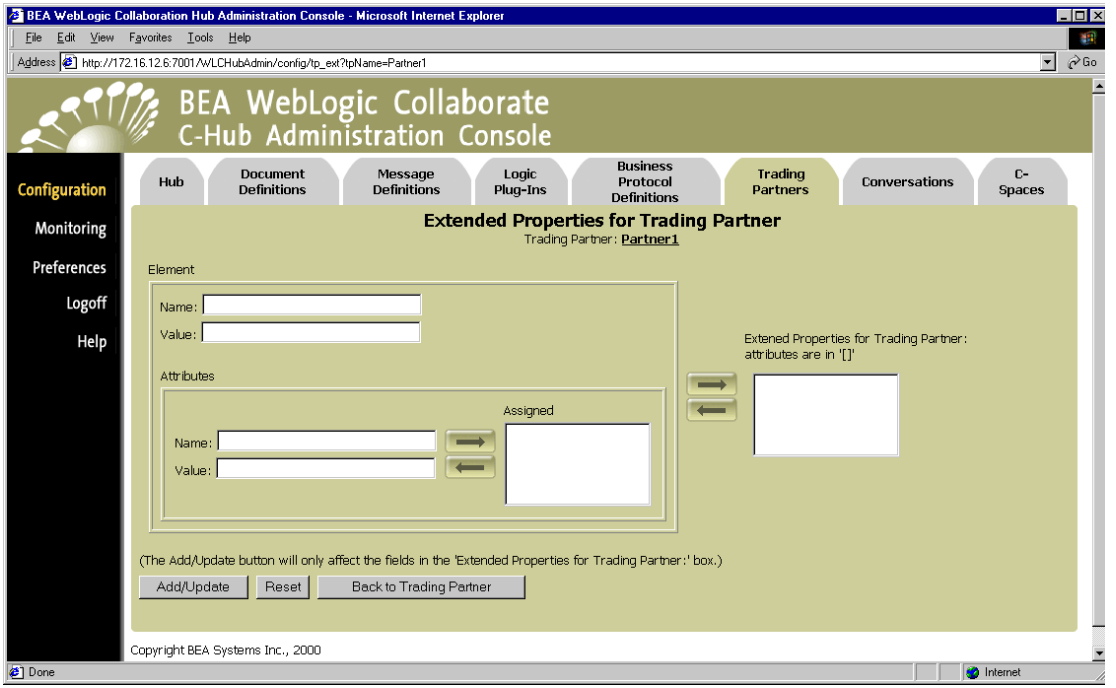
1. From the main Trading Partners tab, click the trading partner to which you want to add extended attributes.

   This brings up a screen showing the configuration details for the selected trading partner.

2. Click on the **Add/Update Extended Properties** button.

   This brings up the Extended Properties screen for the selected trading partners.

Figure 14-5 Extended Properties for a Trading Partner



3. Fill in the fields as described in the following table.

Table 14-5 TP Protocol Assignment Fields

Field	Description
Element Name	Indicate the name of the element for the extended property.
Element Value	Define a text string that you want contained in the element.
Attributes	Define the list of available attributes by first defining a name/value attribute pair for the extended property in the fields **Attributes Name** and **Attributes Value**. Then use the right arrow button to move the name/value pair into the *Available Attributes* list, which is used to construct the extended properties. Repeat this process to build a list of available attributes. (Use the right facing arrow to move attributes into this list; use the left facing arrow to remove attributes from the list.)
Extended Properties for Trading Partner	Use the second set of left/right arrow buttons (on the right) to move attribute name/value pairs from the *Attributes to assign* list into the *Extended Properties for a Trading Partner* list. (Use the right facing arrow to move attributes into the extended properties list; use the left facing arrow to remove attributes from the extended properties list.)

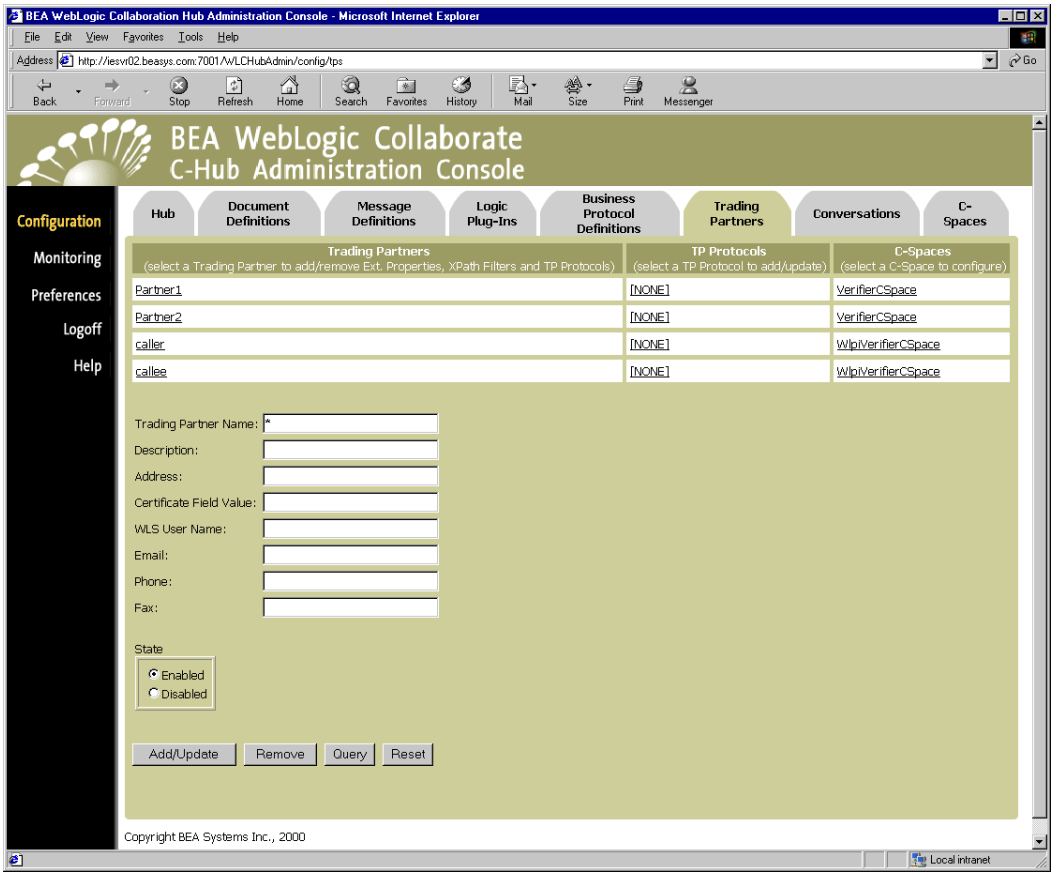
4. Click **Add/Update** to update the trading partner to use the extended properties you added or modified.

Viewing Trading Partners on a C-Hub

To view trading partners configuration information:

1. Click **Configuration** in the left navigation bar.
2. Click the **Trading Partners** tab to display the Trading Partners configuration screen.

Figure 14-6 Trading Partners Configuration



The Trading Partners configuration screen shows a table of trading partners currently defined in the c-hub, the Trading Partner (TP) protocols to which they are assigned (if any), and the c-spaces to which the trading partners belong.

Using the Query Feature to Find Trading Partners

Notice the * in the Trading Partner Name field. This is a special wildcard character used for queries. In the above screen, the table shows all the trading partners on the c-hub. You can type text strings in the Trading Partner Name field and do queries on them to refine the table display. For instance, to display just trading partners that start with "Pa", you can input `Pa*` in the System-ID field and click **Query**. (The query is case-sensitive.) To display information just for `Partner1`, you can enter `Partner1` in the Trading Partner Name field and click **Query**, or just click on the `Partner1` name in the *Trading Partners* list.

Modifying an Existing Trading Partner

To modify an existing trading partner:

1. Click **Configuration** in the left navigation bar.
2. Click the **Trading Partners** tab to display the Trading Partners configuration screen.
3. In the *Trading Partners* list, click on the trading partner you want to modify. The configuration information for that trading partner is displayed.
4. Update the fields you want to modify as described in Table 14-1.
5. Click **Add/Update** to update the trading partner.

   (You can also click **Reset** to discard your current changes and start again.)

Viewing Configuration Details for Trading Partners

To display details for a specific trading partner from the Trading Partners configuration tab, click on its name in the *Trading Partners* list.

How Do Trading Partners Relate to C-Spaces?

For information on how to add a trading partner to a c-space, see Chapter 16, "Working with C-Spaces."

Viewing Details about the C-Space to which a Trading Partner Belongs

For trading partners that have already been added to c-spaces, you can view the configuration details of the related c-space. To display details of the c-space a trading partner belongs to from the Trading Partners configuration tab, click on the c-space name in the *C-Spaces* column.

Removing a Trading Partner

To remove a trading partner:

1. Click **Configuration** in the left navigation bar.
2. Click the **Trading Partners** tab to display the Trading Partners configuration screen.
3. In the Trading Partner Name field, enter the full name of the trading partner that you want to remove, and click **Remove**.

   or

   Click on the trading partner you want to remove to display the configuration details for that trading partner. On this screen, click **Remove**.

   A confirmation dialog is displayed. If you click **OK**, the specified trading partner is removed and you are returned to the main Trading Partners configuration screen.

Note: If the object you remove *references* or *is referenced by* other objects, those references will be removed. If the object you remove *contains* other objects, those objects will be removed. Removing a trading partner should not affect other c-hub entities. See Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

Monitoring Trading Partners

For information on how to view and monitor detailed information on existing trading partners, see Monitoring Trading Partners on the C-Hub in Chapter 18, “Monitoring the C-Hub.”

CHAPTER

15 Setting Up Conversations

The following sections provide key concepts and procedures for configuring and working with conversations on the C-Hub Administration Console:

- What Is a Conversation?
- Setting Up a Conversation
- Viewing Conversations on a C-Hub
- Removing a Conversation
- Monitoring Conversations

For introductory information about conversations, see Overview in *BEA WebLogic Collaborate Getting Started*. For information about developing trading partner applications that initiate and participate in conversations, see the *BEA WebLogic Collaborate Developer Guide*.

What Is a Conversation?

A *conversation* is a series of predefined message exchanges between collaborators/trading partners that take place in a c-space in the context of a predefined business model. Each message in the conversation may cause any number of back-end transactions to occur depending on the actions implemented by the trading partner in its local business protocol definition. Trading partners interact with each other in conversations.

For a complete explanation of conversations and other related concepts, see *BEA WebLogic Collaborate Getting Started*, particularly the Overview.

Setting Up a Conversation

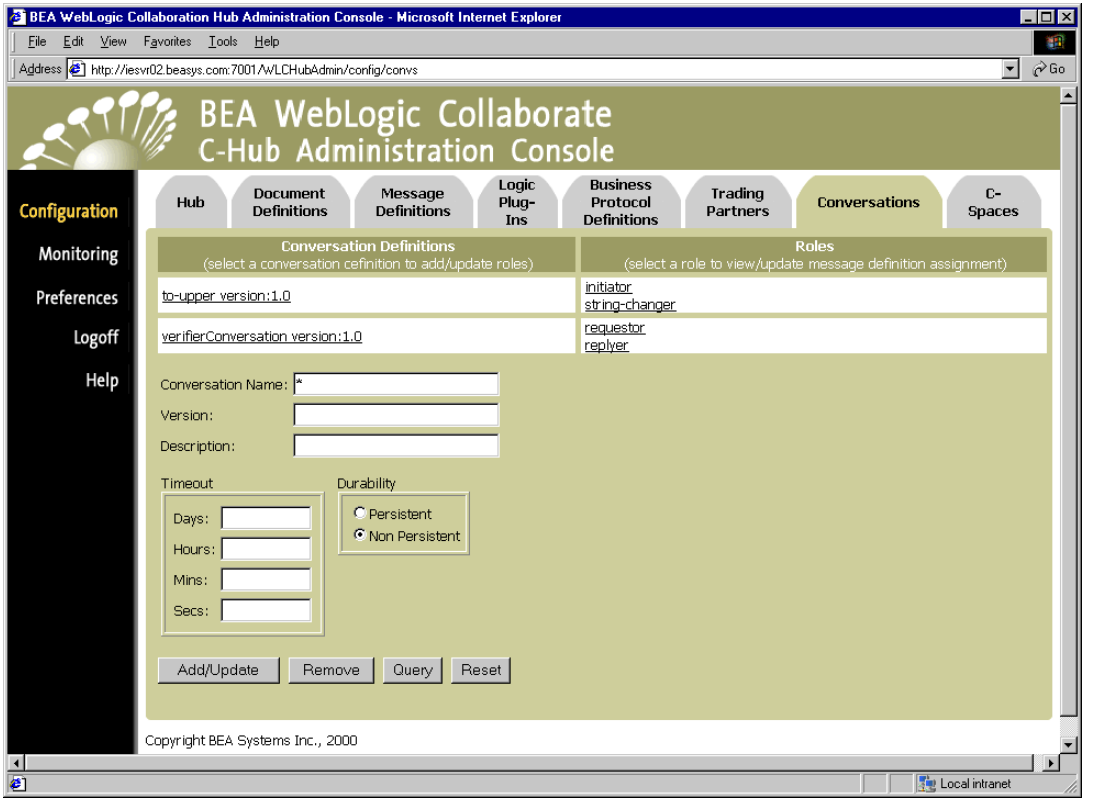
Setting up a new conversation on the c-hub consists of the following tasks:

- Step 1. Name and Describe the Conversation.
- Step 2. Assign Roles to a Conversation.
- Step 3. Assign Message Definitions to Roles.

Step 1. Name and Describe the Conversation.

1. Click **Configuration** in the left navigation bar.
2. Click the **Conversations** tab to display the Conversations configuration screen.

Figure 15-1 Conversations Configuration



3. Fill in the fields as described in the following table.

Table 15-1 Conversations Fields

Field	Description
Conversation Name	Name for the conversation (Limit is 256 characters.)
Version	Version number of the conversation (Limit is 80 characters.)
Description	A brief description of the conversation (Limit is 256 characters.) (optional)
Timeout	Indicate the amount of time a conversation remains valid (in days, hours, minutes and seconds). After this amount of time, the c-hub will terminate the conversation even if there are outstanding messages. (optional)
Durability	Select **Persistent** (stored and recoverable) or **Non-Persistent**.

4. At this point if you click **Add/Update**, the new conversation will show up in the Conversation Definitions list, but no roles will be assigned to it.

   (You can also click **Reset** to discard your current changes and start again.)

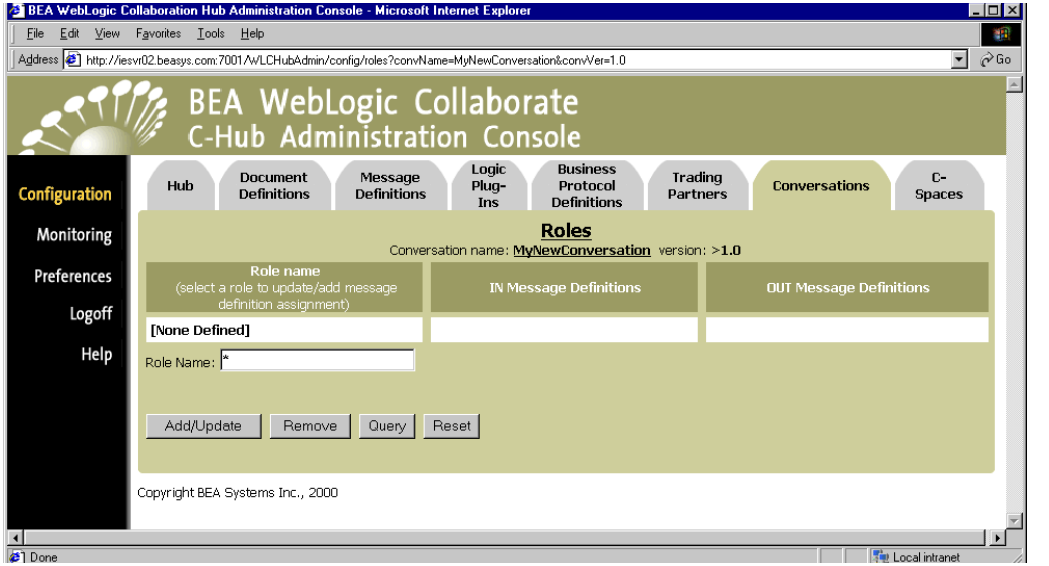
The next step is to assign roles.

Step 2. Assign Roles to a Conversation.

To assign roles to a conversation, click on the role under the *Roles* list on the main screen. (If you haven't assigned a role, you can click on [NONE], which shows up under *Roles* for that conversation.)

This brings up the screen where you can assign roles to the selected conversation.

Figure 15-2 Assigning Roles to a Conversation



This screen shows a list of roles and related message definitions currently assigned to the selected conversation. From here you can remove or update the roles currently assigned (if any) and also assign new roles to a conversation.

To assign a new role:

1. Enter the name of the role in the Role Name field as described in the table below.

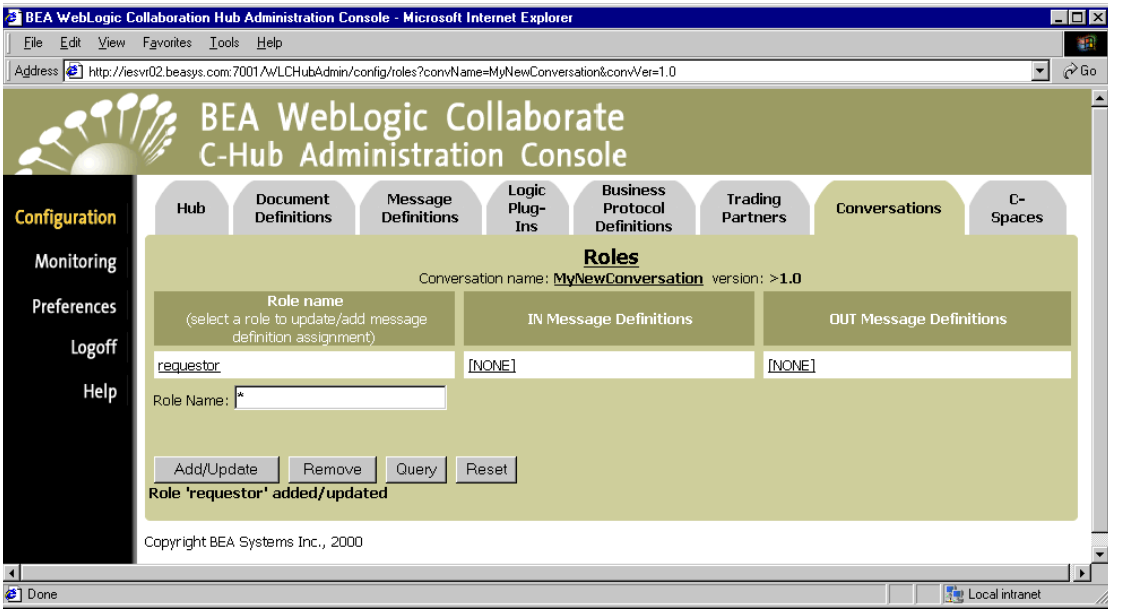
Table 15-2 C-Spaces Fields

Field	Description
Role Name	Enter the name of the role you want to create. (Limit is 256 characters.)

2. Click **Add/Update** to add the new role.

   You will be returned to the main Roles screen for the selected conversation and your new role will show up in the listing of roles. When you add a new role to a conversation, the IN and OUT message definitions lists will display [NONE] for that role.

Figure 15-3 Newly Added Role



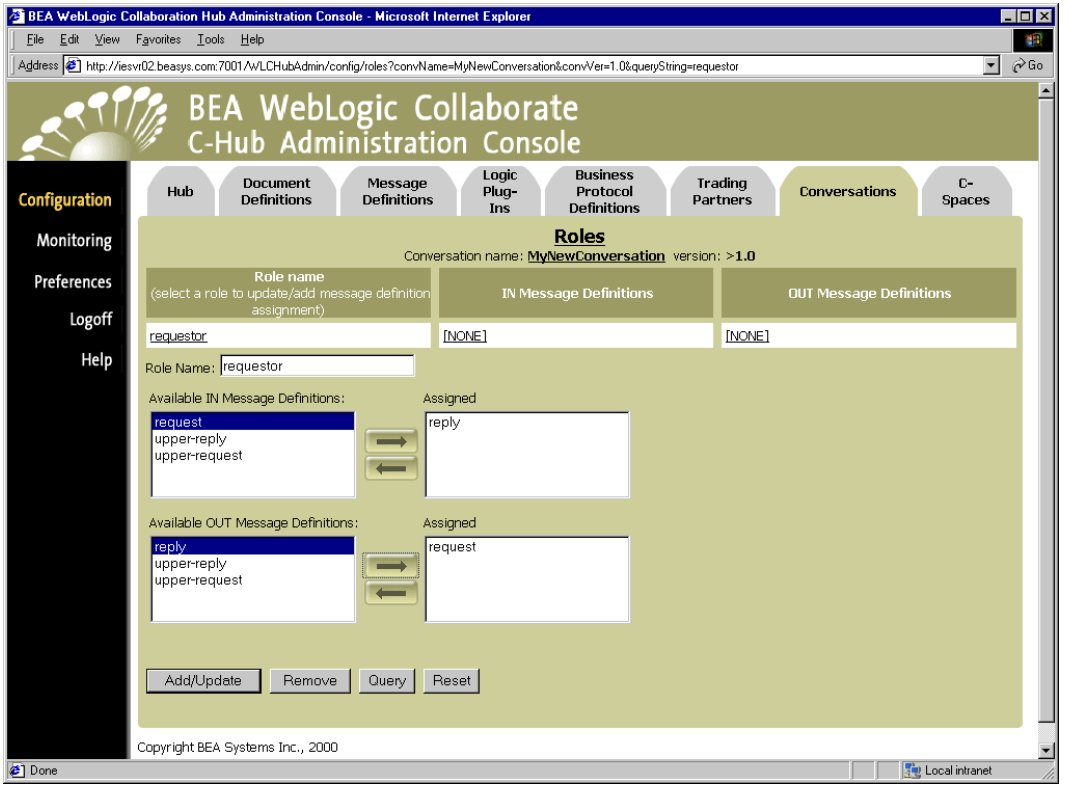
The next step is to assign message definitions to the role.

Note: To update an existing role, simply click on the role name in the *Role name* column.

Step 3. Assign Message Definitions to Roles.

You can bring up the Roles screen for a particular conversation by clicking on a role name in the *Roles* list for that conversation (in the Conversations configuration tab shown on Figure 15-1). You can also click on the [NONE] under the Message Definitions lists if no message definitions are defined yet for that role.

Figure 15-4 Assigning Message Definitions to Roles in a Conversation



This screen shows the message definitions (defined in the Message Definitions tab) that are currently (if any) assigned to a role.

The Role Name field below the columns shows the role you just selected and to which you will assign message definitions.

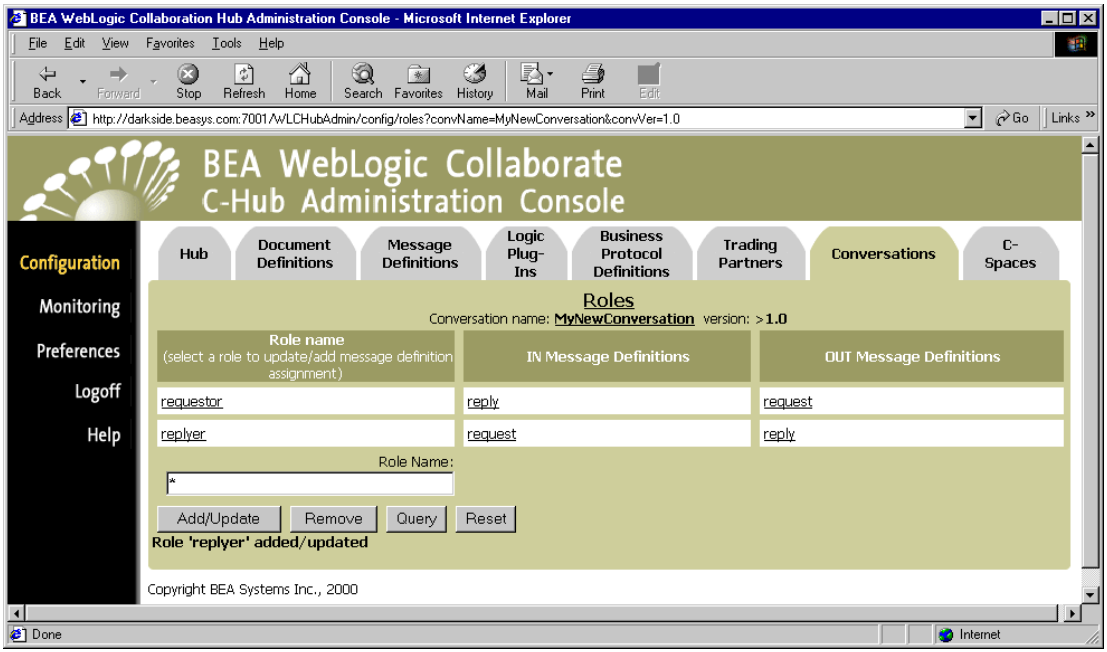
Make sure you have the roles for the *conversation* you want to work with. The conversation name shows at the top of the screen.

Use the arrow buttons to move the message definitions you want to assign to the selected conversation into the *Assigned* lists for both IN and OUT message definitions. An IN message definition is a message definition the trading partner in this role will receive. An OUT message definition is one that the trading partner in this role will send. (The IN and OUT *Message Definitions* show the list of available message definitions. The *Assigned* list represents the message definitions that are actually assigned to the selected role.)

When you have the appropriate message definitions listed in the *Assigned* lists, click **Add/Update** to assign the message definitions to the selected role in the conversation.

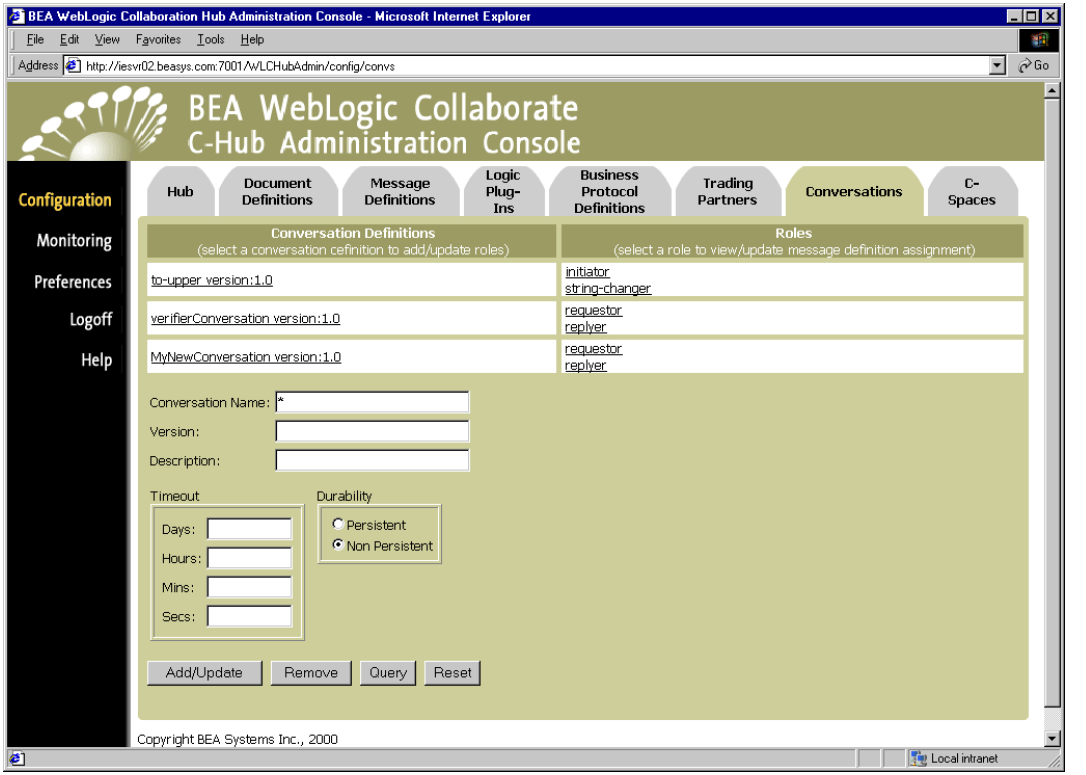
The new role and message definition assignments you added for that conversation are displayed.

Figure 15-5 Role/Message Definition Assignments for a Conversation Succeeded



You will also see your changes reflected in the full list of conversations on the main Conversations tab. (To see this, click the **Conversations** tab.)

Figure 15-6 New Role with Message Definitions Shown on Main Conversations Tab

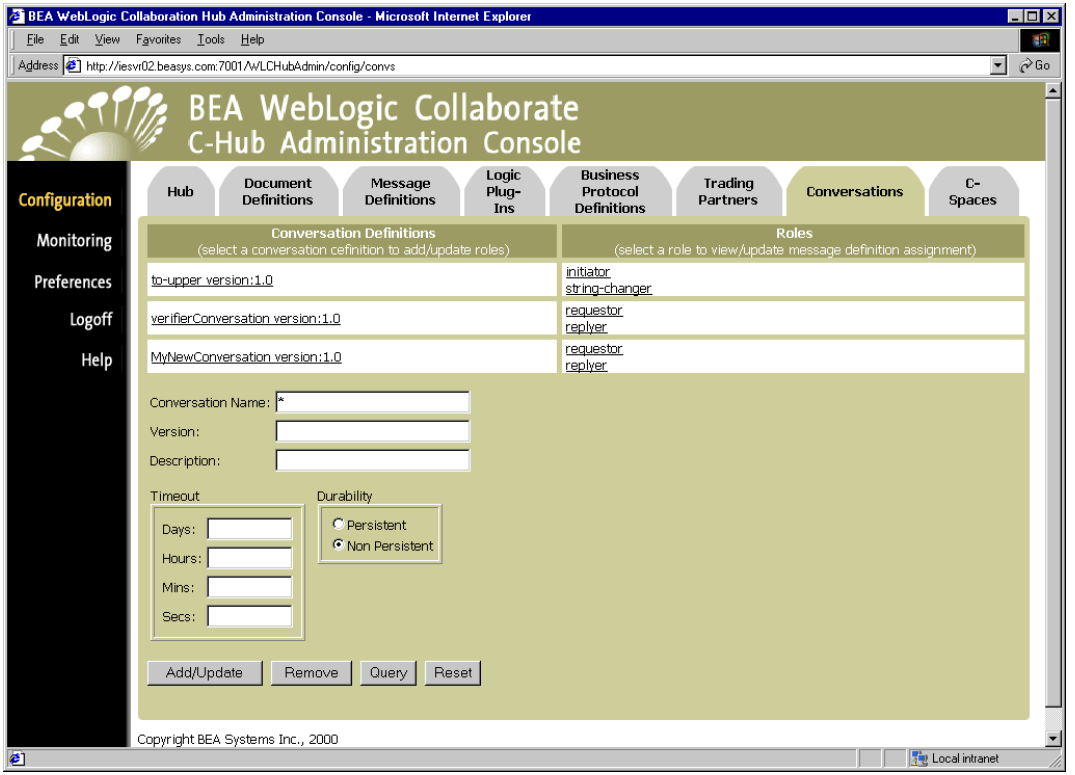


Viewing Conversations on a C-Hub

To view the conversations on a c-hub:

1. Click **Configuration** in the left navigation bar.
2. Click the **Conversations** tab to display the Conversations configuration screen.

Figure 15-7 Viewing Conversations



The conversations configuration screen shows a table of conversations currently defined in the c-hub and the roles that are set up to participate in those conversations.

Using the Query Feature to Find Conversations

Notice the * in the Conversation Name field. This is a special wildcard character used for queries. In the above screen, the table shows all the conversations on the c-hub. You can type text strings in the Conversation Name field and do queries on them to refine the table display. For instance, to display just conversations that start with "Co", you can input `Co*` in the Conversation Name field and click **Query**. (The query is case-sensitive.) To display information just for `Conversation1`, you can enter `Conversation1` in the Conversation Name field and 1.0 in the version field and click **Query**, or just click on the `Conversation1` name in the *Conversation Definitions* list.

Removing Conversations and Roles

You can remove a role in a conversation, or remove the entire conversation at once, including its roles:

- Removing a Role from a Conversation
- Removing a Conversation

Removing a Role from a Conversation

Roles are always associated with a particular conversation. To remove a role from a conversation:

1. Click **Configuration** in the left navigation bar.
2. Click the **Conversations** tab to display the Conversations configuration screen.
3. Find the conversation from which you want to remove a role, and click on the associated role in the Roles list.

   This brings up the detail configuration screen for the selected role associated with a particular conversation, as indicated at the top of the screen.

4. Click **Remove**.

   If the object you are about to remove is referenced by other object configurations (for example, if a c-space is using a conversation containing the role you are about to remove), then the confirmation dialog indicates this and asks if you want to continue. Click **OK** if you want to continue with the Remove.

   The specified role is removed, and you are returned to the main Roles configuration screen for the particular conversation you are working with. Other roles may still be listed for that conversation.

Removing a Conversation

Roles exist within conversations, so keep in mind that if you remove a conversation, you automatically remove any roles that are included in (assigned to) that conversation.

To remove a conversation:

1. Click **Configuration** in the left navigation bar.
2. Click the **Conversations** tab to display the Conversations configuration screen.
3. In the Conversation Name field, enter the full name of the conversation that you want to remove, and click **Remove**.

   or

   Click on the conversation you want to remove to display the configuration details for that conversation. On this screen, click **Remove**.

   A confirmation dialog is displayed. If you click **OK**, the specified conversation is removed and you are returned to the main Conversations configuration screen.

Note: If the object you remove *references* or *is referenced by* other objects (for example, if a c-space is using the conversation you are about to remove), those references will be removed. If the object you remove *contains* other objects, those objects will be removed. See Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

Monitoring Conversations

For information on how to view and monitor detailed information on existing conversations, see Monitoring Conversations on the C-Hub in Chapter 18, “Monitoring the C-Hub.”

CHAPTER

16 Working with C-Spaces

The following sections provide key concepts and procedures for configuring c-spaces on the C-Hub Administration Console:

- What Is a C-Space?
- Creating a New C-Space
- Adding New Roles
- Viewing C-Spaces on a C-Hub
- Modifying an Existing C-Space
- Removing a C-Space
- Monitoring C-Spaces

For introductory information about c-spaces, see Overview in *BEA WebLogic Collaborate Getting Started*. For information about developing c-space monitoring applications, see Developing Management Applications in the *BEA WebLogic Colaborate Developer Guide*.

What Is a C-Space?

A collaboration space (c-space) is a container for predefined trading partners that conduct and coordinate conversations with each other. A collaboration space supports the following:

- A single business model
- A business message vocabulary

- A registered set of trading partners

An e-market owner (who hosts a c-hub) can support any number of collaboration spaces on the Web concurrently. Each c-space can support any number of trading partners. Each trading partner can subscribe to business messages, which are communicated through a mediated messaging system, and can define local workflow and actions to execute upon receipt of business events (which are messages or documents).

For a complete explanation of c-spaces and other related concepts, see *BEA WebLogic Collaborate Getting Started*, particularly the Overview.

Creating a New C-Space

Creating a new c-space on a c-hub consists of the following tasks:

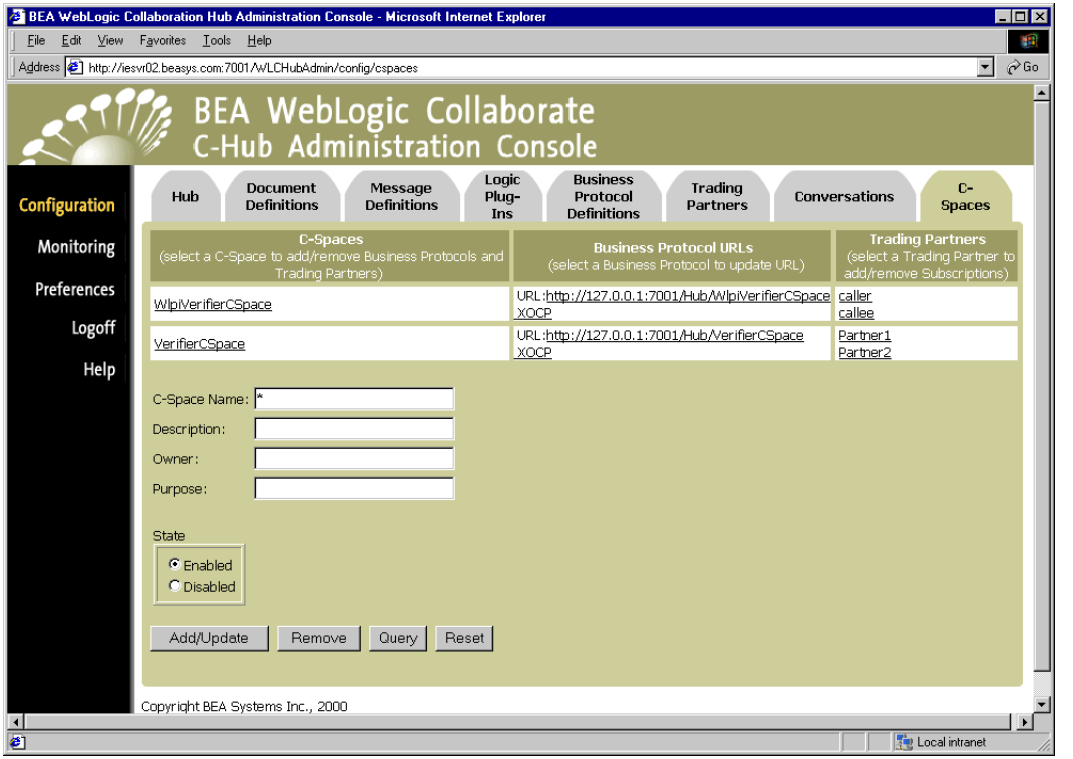
- Step 1. Name and Describe the C-Space.
- Step 2. Assign a Business Protocol for a C-Space.
- Step 3. Add Trading Partners to a C-Space.
- Step 4. Assign Subscriptions (Roles and Conversations) to a Trading Partner.

Step 1. Name and Describe the C-Space.

To create a new c-space on a c-hub:

1. Click on **Configuration** in the left navigation bar.
2. Click on the **C-Spaces** tab to display the C-Spaces configuration screen.

Figure 16-1 C-Spaces Configuration



3. Fill in the fields as described in the following table.

Table 16-1 C-Spaces Fields

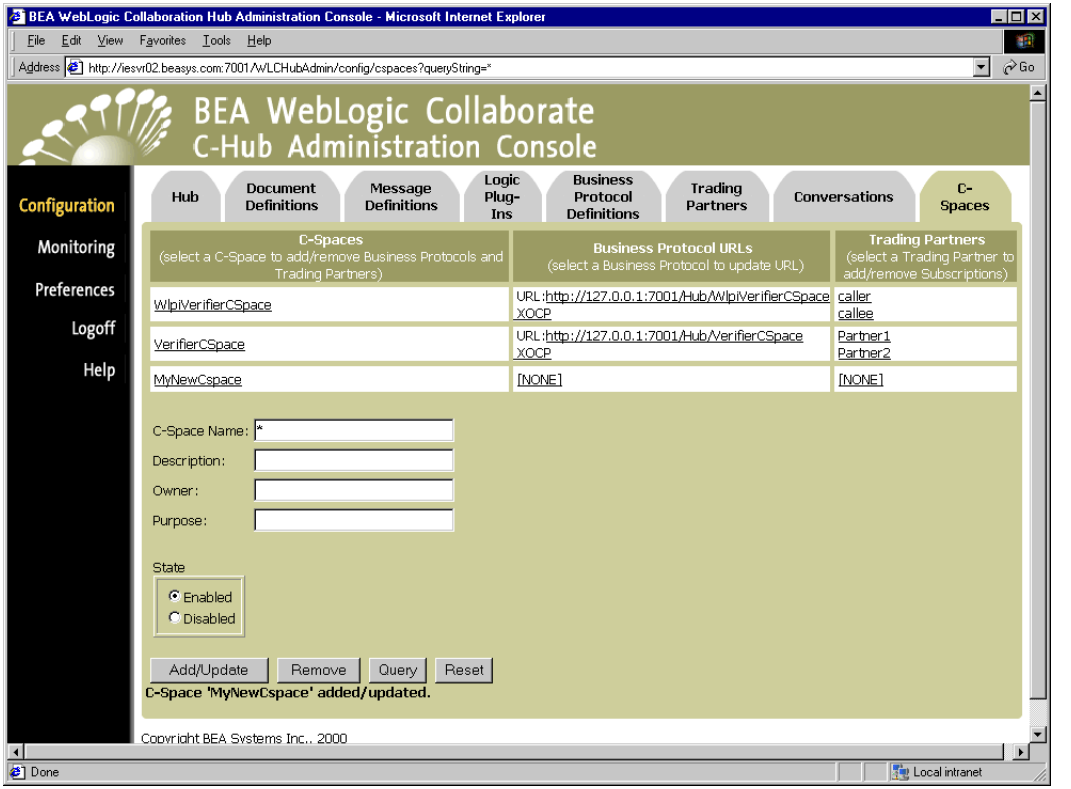
Field	Description
C-Space Name	Enter the name of a c-space you want to create or modify. (Limit is 256 characters.)
Description	Enter a brief description of the c-space (Limit is 256 characters.) (optional)
Owner	Enter the owner of the c-space (Limit is 80 characters.) (optional—for description only; does not correspond to access control)
Purpose	Enter a brief description of the purpose of the c-space (Limit is 256 characters.) (optional)
State	■ Select **Enabled** if you want the c-space to accept new conversations (if started). ■ Select **Disabled** if you do not want the c-space to accept new conversations.

4. If you click on **Add/Update**, the c-space will be created but no trading partners will be assigned to it.

   Your new c-space should show up in the *C-Spaces* list.

   (You can also click **Reset** to discard your current changes and start again.)

Figure 16-2 New C-Space Added



When you first add a new c-space, [NONE] will show up under both the Business Protocol URLs and the Trading Partners lists for that c-space. You need to assign both of these. The next step is to assign a *business protocol* to use in the new c-space.

Step 2. Assign a Business Protocol for a C-Space.

Assigning a business protocol to a c-space entails the following:

- Providing a location (URL) where the c-space can receive incoming messages

  Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol.

  Warning: A URL for a c-space/business protocol combination can be used only by the c-hub and c-enablers. If customer-supplied software uses one of these URLs, messages will not be processed correctly.

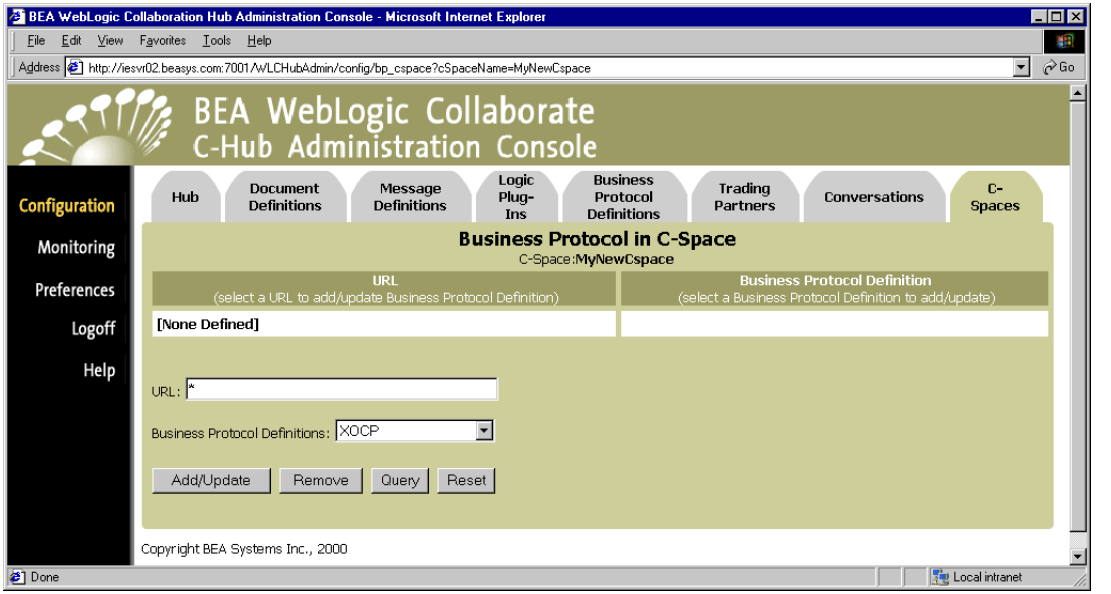
- Indicating a business protocol definition with code that specifies how the c-space will process messages

To assign a business protocol for the c-space:

1. Click on the [NONE] under Business Protocols for the c-space you want to modify.

   This brings up the Business Protocol configuration screen for the selected c-space.

Figure 16-3 Assigning a Business Protocol for a C-Space



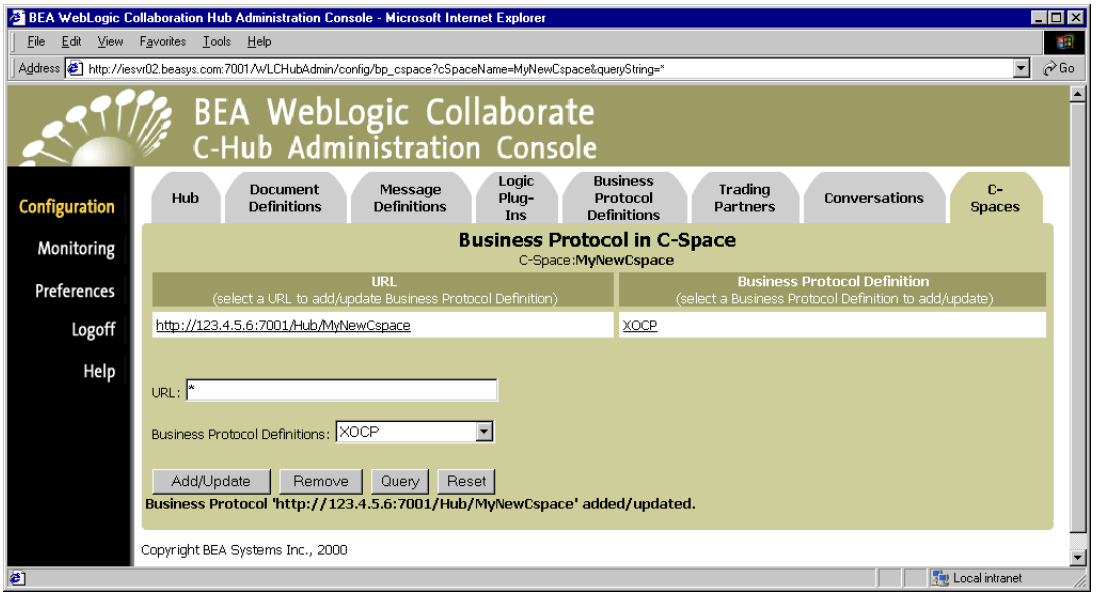
2. Fill in the fields as described in the following table.

Table 16-2 Business Protocols for C-Space Fields

Field	Description
URL	Enter a URL for the location of the business protocol definition in this c-space. Each c-space/business protocol combination has a unique URL. A trading partner uses this URL to access a particular c-space using a particular business protocol. **Warning:** A URL for a c-space/business protocol combination can be used only by the c-hub and c-enablers. If customer-supplied software uses one of these URLs, messages will not be processed correctly.
Business Protocol Definitions	Select the business protocol definition from the drop-down list.

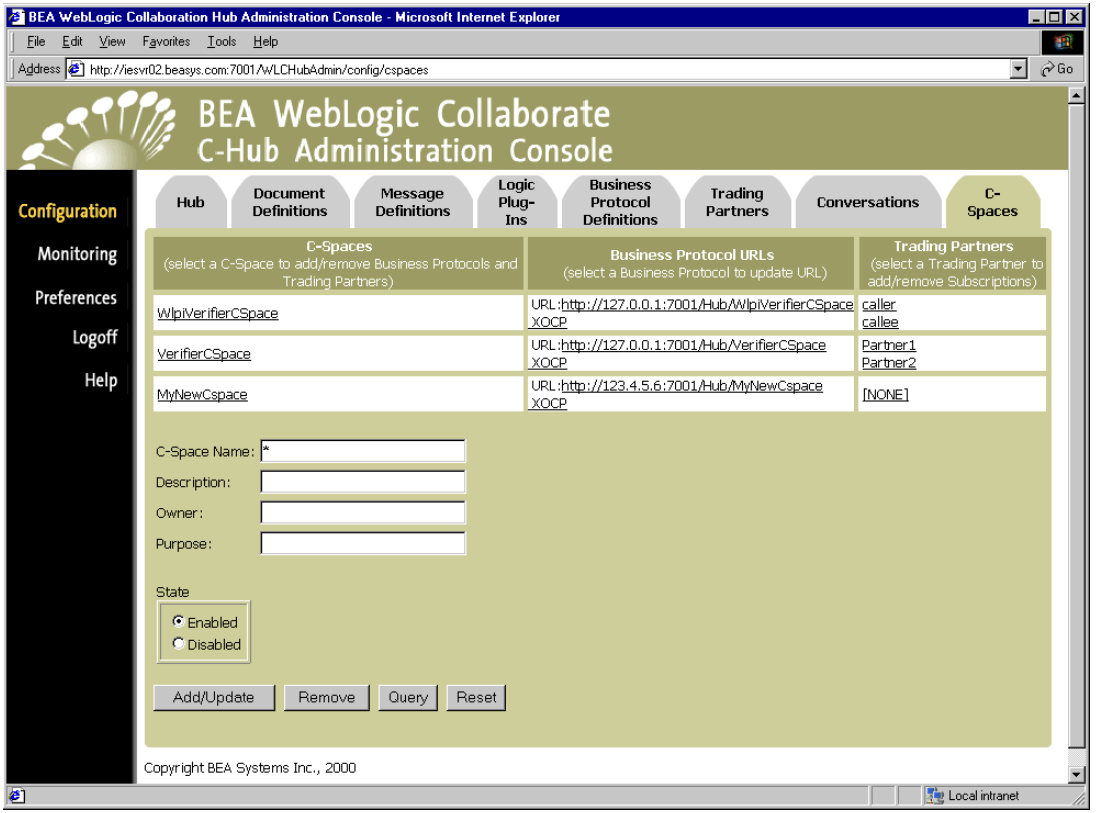
3. Click **Add/Update** to add the new business protocol definition to the selected c-space.

Figure 16-4 Business Protocol Definition Added to a C-Space



If you now click the **C-Spaces** tab again to get back to the C-Spaces screen, you should see the business protocol you assigned reflected here.

Figure 16-5 Main C-Spaces Tab Showing Business Protocol Assigned for the New C-Space

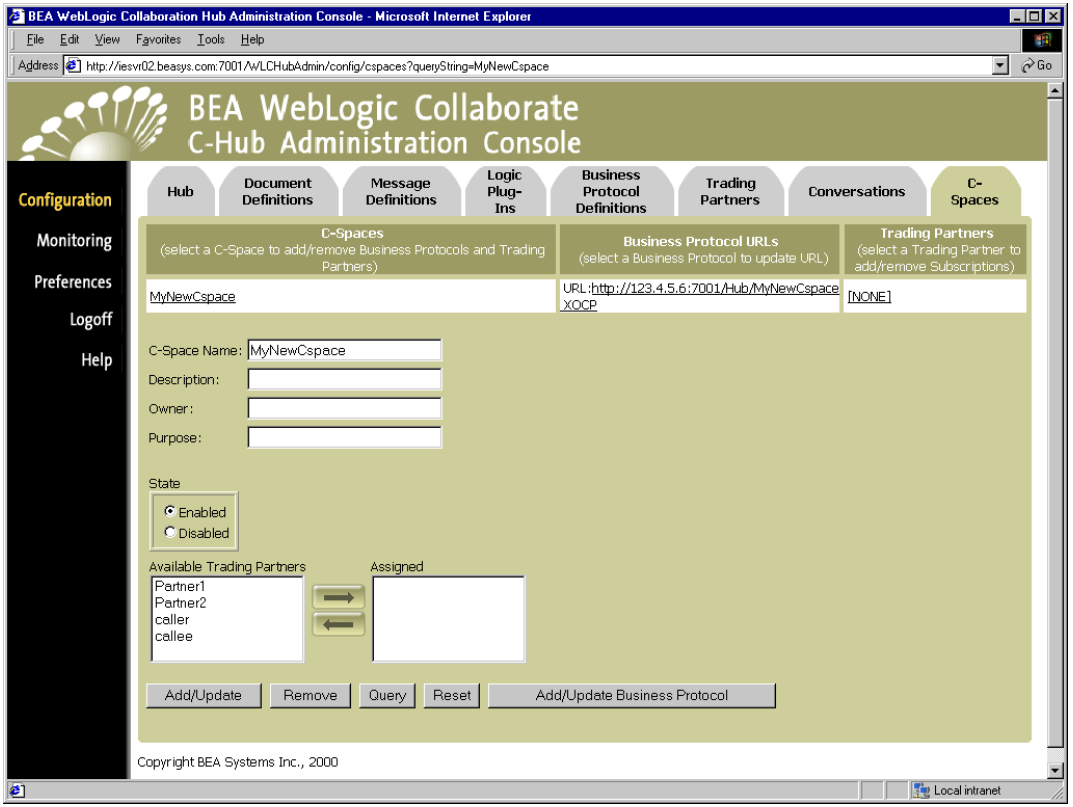


Notice that now only the Trading Partners list shows [NONE] for the new c-space. The final step is to add trading partners to the c-space.

Step 3. Add Trading Partners to a C-Space.

To add trading partners to a c-space, make sure you are on the C-Spaces configuration tab, then click on the name of the c-space in the *C-Space* list. (You can also click on the [NONE] under Trading Partners for the c-space you want to modify.) A detailed configuration screen for that c-space is displayed.

Figure 16-6 Adding Trading Partners to a C-Space

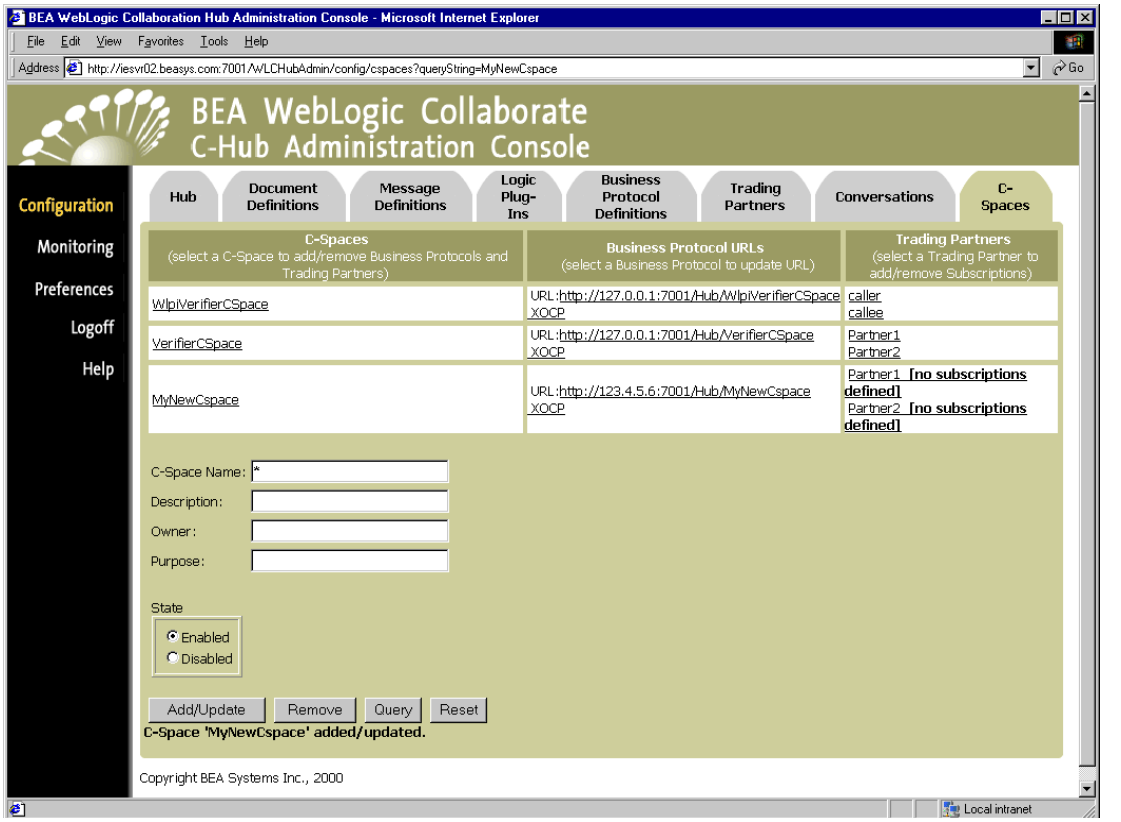


You can assign trading partners to the selected c-space by using the arrow keys to move the desired partners into or out of the *Assign to C-Space* list. (The *Trading Partners* list provides a list of available trading partners. The *Assign to C-space* list represents the trading partners assigned to the selected c-space.)

When you have the appropriate trading partners listed under *Assign to C-space*, click **Add/Update** to assign the partners to the c-space.

The selected partners should show up under Trading Partners assigned to the appropriate c-space on the main C-Spaces configuration screen.

Figure 16-7 Trading Partners Added to C-Space but Not Yet Subscribed to a Conversation

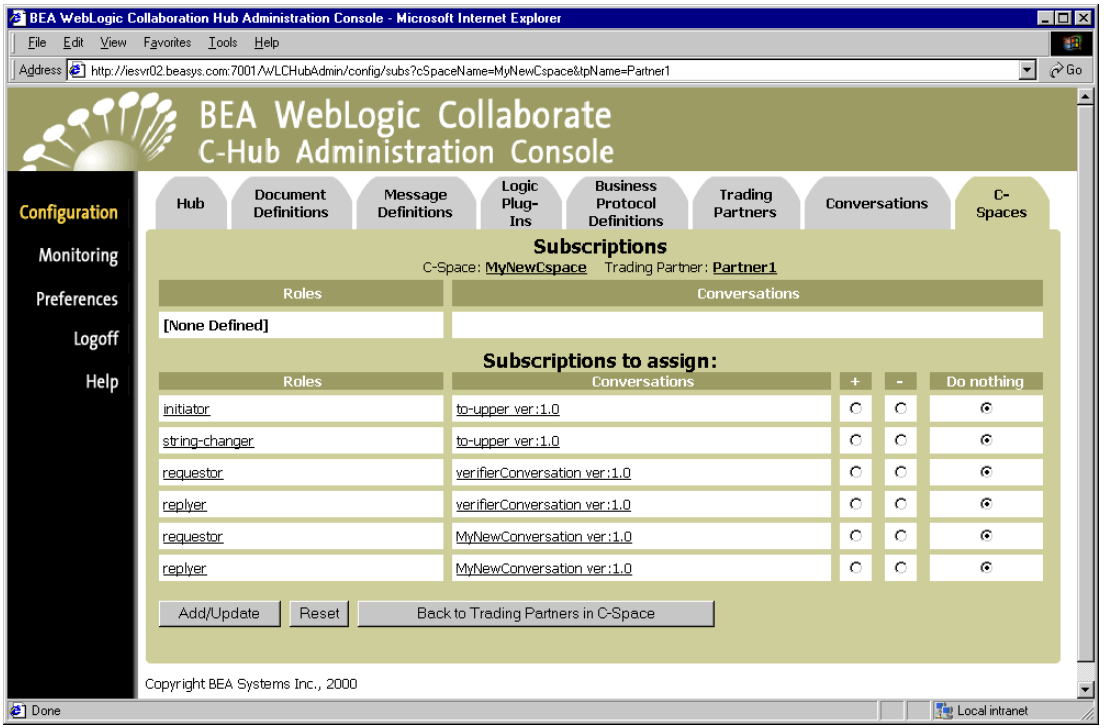


Note that the trading partner definition is not complete at this point, as indicated by the [no subscriptions defined] message next to the trading partner name in the Trading Partners list. The next steps are to create some roles and then subscribe the trading partners to play defined roles in specific conversations.

Step 4. Assign Subscriptions (Roles and Conversations) to a Trading Partner.

To assign a subscription to a trading partner, make sure you are on the C-Spaces configuration tab, then click on the name of the trading partner in the Trading Partners list. The C-Spaces "Subscriptions" information for the selected trading partner is displayed. Here, you can assign subscriptions to that trading partner. (At the top of the screen, notice the name of the c-space and the name of the trading partner for which you are about to modify subscriptions.)

Figure 16-8 Subscriptions for a Trading Partner in a C-Space



This screen allows subscriptions to be assigned. The bottom table, labeled *Subscriptions to assign*, are possible subscriptions available (based on the conversation defined earlier). These are not subscriptions that exist in the c-space, only the possible choices. They only exist when assigned to a trading partner (shown in the top table, labeled *Subscriptions*).

The first table shows the subscriptions currently assigned (if any) to the trading partner.

The second table shows the possible subscriptions that may be assigned to the trading partner. The possible subscriptions are based on information that was entered for the conversation definition (in the Conversation tab). The radio buttons are:

- `+` : add the subscription to the trading partner (if it is not already assigned to the trading partner)
- `-` : remove the subscription from the trading partner (if it is assigned)
- `Do nothing`: no action for this subscription

When you have selected the appropriate role/conversation subscription you want to assign to this trading partner, click **Add/Update** to apply it. You should see your change reflected in the Subscriptions list at the top of the screen (the first table).

(For example, to assign the roles of requestor in this c-space to the selected trading partner, you would select the '+' radio button for *requestor* and click **Add/Update** to see the assignment to take effect.)

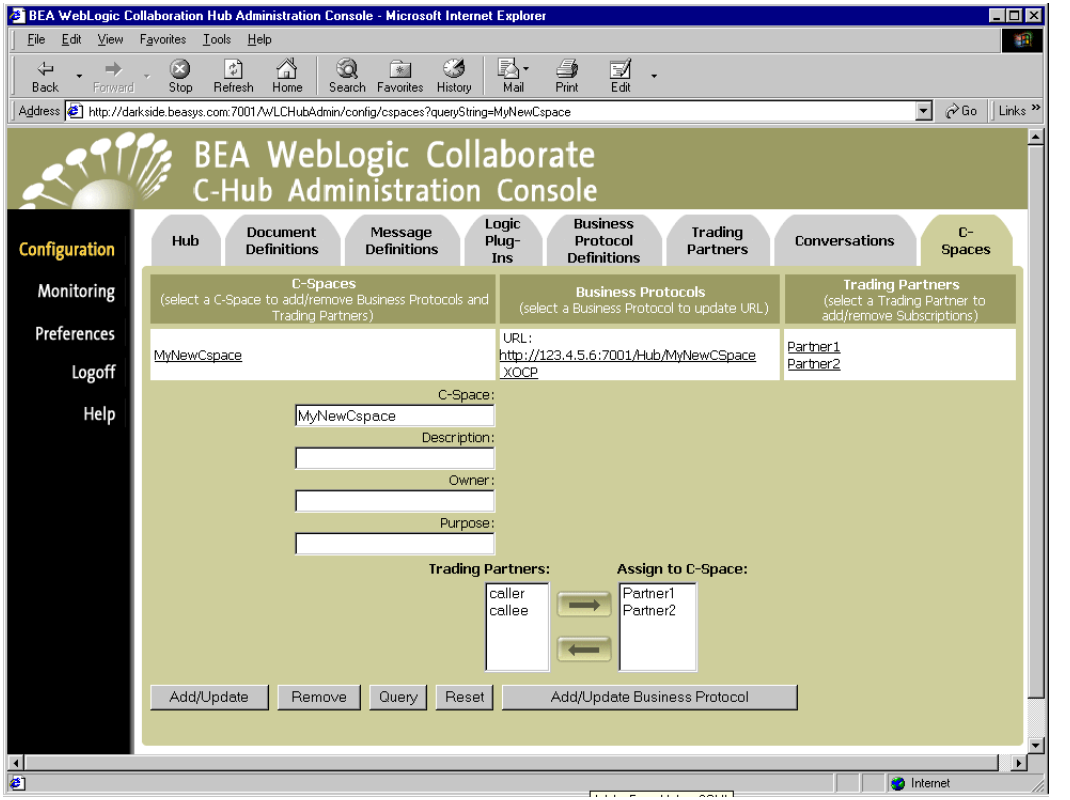
Figure 16-9 Trading Partner Subscribed to a Role in a Conversation

To return to the trading partners screen for this c-space, click the **Back to Trading Partners in C-Space** button. If you subscribed the first partner to a c-space, you should see the subscription show up on the Trading Partners screen.

You can now click on the name of the next trading partner you want to subscribe to a c-space and follow the procedure of adding roles as needed (described above).

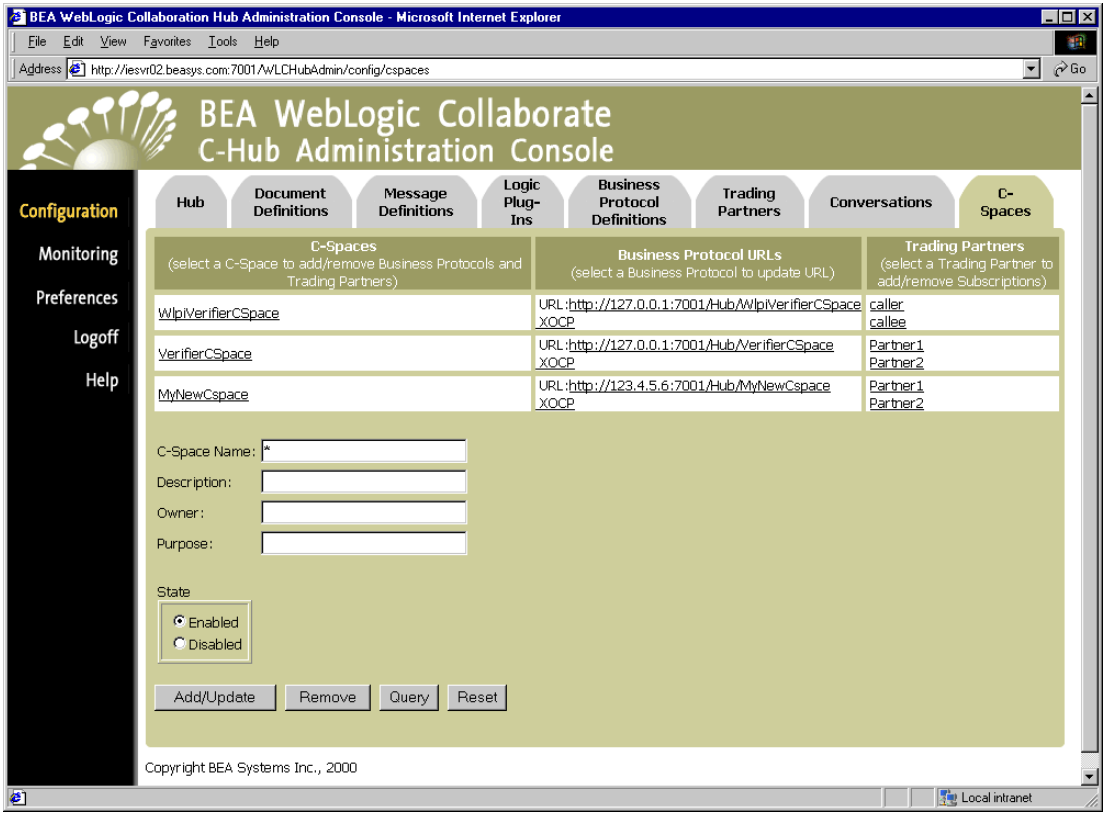
To return back to the trading partners screen for this c-space, click the **Back to Trading Partners in C-Space** button. If you subscribed the second partner to a c-space, you now should see the subscription show up on the Trading Partners and the C-spaces screens.

Figure 16-10 Trading Partners Successfully Added to a C-Space and Subscribed to Roles in a Conversation



If you click on the **C-Spaces** tab again to see the main C-Spaces screen, the new c-space you added should show up with its subscribed trading partners next to it. (The [NONE] and [no subscriptions defined] messages should be gone now if you have added trading partners and subscribed them to conversations.)

Figure 16-11 C-Spaces Main Screen Showing New C-Space Added



Adding New Roles

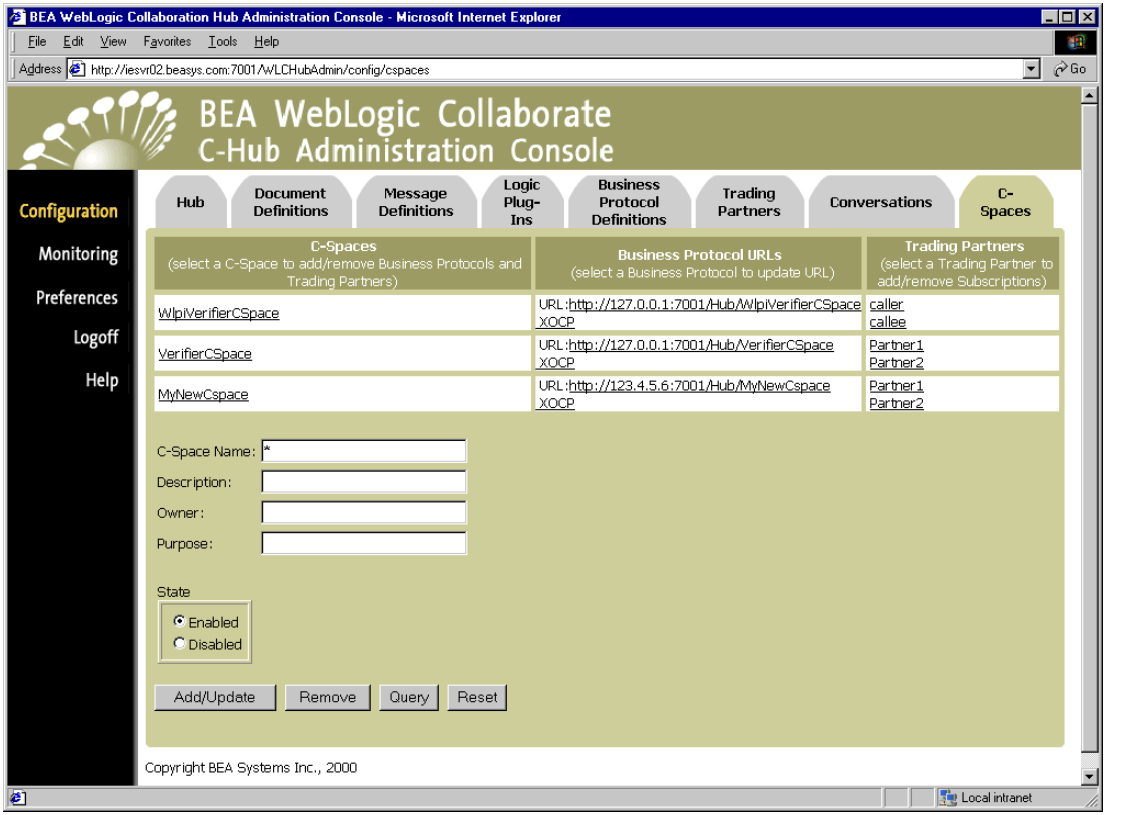
An explanation for how to add new roles is included in “Step 3. Assign Message Definitions to Roles.” on page 15-7 as a part of Chapter 15, “Setting Up Conversations.”

Viewing C-Spaces on a C-Hub

To view the existing c-spaces on a c-hub:

1. Click on **Configuration** in the left navigation bar.
2. Click on the **C-Spaces** tab to display the C-Spaces configuration screen.

Figure 16-12 C-Spaces Configuration



The C-Spaces configuration screen shows the c-spaces currently defined in the c-hub, the business protocols assigned for the c-spaces, and the trading partners that exist in each of the c-spaces.

Using the Query Feature for C-Spaces

Notice the * in the C-Space field. This is a special wildcard character used for queries. In the above screen, the table shows all the c-spaces on the c-hub. You can type text strings in the C-Space field and do queries on them to refine the table display. For instance, to display just c-spaces that start with "CS", you can input `CS*` in the C-Space field and click on **Query**. (The query is case-sensitive.) To display information just for `CSpace1`, you can enter `CSpace1` in the C-Space field and click **Query**, or just click on the `CSpace1` name in the *C-Space* list.

Modifying an Existing C-Space

To modify an existing c-space, make sure you are on the C-Spaces configuration tab, then click on the name of a particular c-space in the C-Space list. To update the subscriptions, click on the trading partner in the Trading Partners list to get to the Subscriptions information.

Removing a C-Space

Warning: Keep in mind, if you remove a c-space you are removing the definition from the WebLogic Collaborate repository. This data is not automatically backed up and therefore may not be recoverable.

To remove a c-space:

1. Click **Configuration** in the left navigation bar.
2. Click the **C-Spaces** tab to display the C-Spaces configuration screen.

3. In the C-Space Name field, enter the full name of the c-space that you want to remove, and click **Remove**.

   or

   Click on the c-space you want to remove to display the configuration details for that c-space. On this screen, click **Remove**.

   A confirmation dialog is displayed. If you click **OK**, the specified c-space is removed and you are returned to the main C-Spaces configuration screen.

Note: If the object you remove *references* or *is referenced by* other objects (for example, if a trading partner is assigned to the c-space you are about to remove), those references will be removed. If the object you remove *contains* other objects, those objects will be removed. See Table 8-2 for details on relationships and dependencies that can exist among configured c-hub objects.

Monitoring C-Spaces

For information on how to view and monitor detailed information on existing c-spaces, see Monitoring C-Spaces on the C-Hub in Chapter 18, “Monitoring the C-Hub.”

CHAPTER

17 Setting Preferences

The following sections provide information about setting the “hidden logic plug-ins” preference on the C-Hub Administration Console:

- About Hidden Logic Plug-Ins
- Hiding or Showing “Hidden” Logic Plug-Ins

About Hidden Logic Plug-Ins

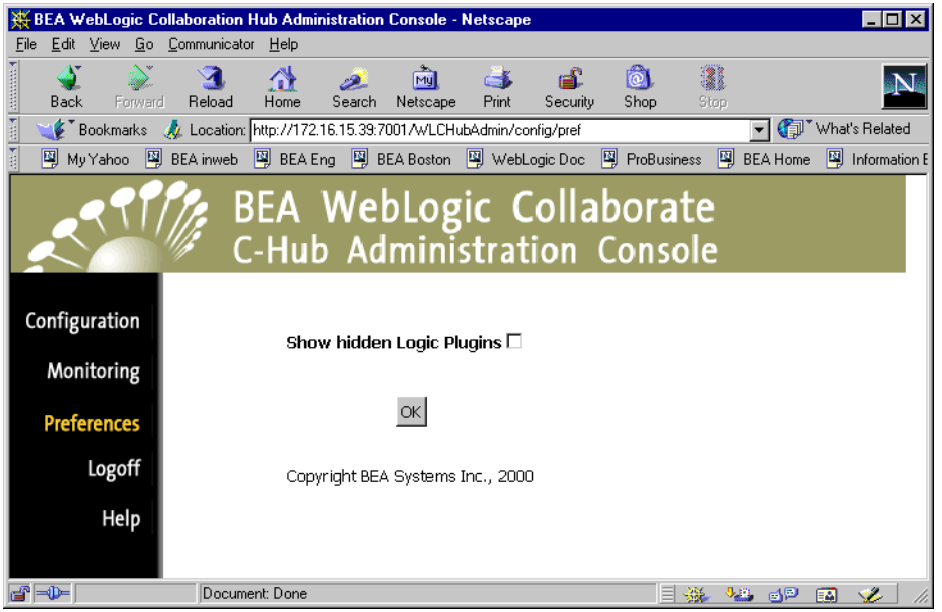
As a c-hub administrator, you have the option of hiding certain logic plug-ins on the C-Hub Administration Console. You can set a parameter on any logic plug-in named `bea.hidden` and set that parameter to a value of `true`. You can then use the c-hub “Preference” setting to hide or show the plug-ins with this setting. Note that the BEA-provided encoder and decoder logic plug-ins for XOCP and RosettaNet are configured with the `bea.hidden=true` parameter, so hiding logic plug-ins through this “Preference” setting automatically hides these BEA plug-ins on the administration console.

Hiding or Showing “Hidden” Logic Plug-Ins

To hide or show hidden logic plug-ins:

1. Click **Preferences** in the left navigation bar to display the Preferences configuration screen.

Figure 17-1 Preferences Configuration



2. Click on the Show hidden Logic Plug-Ins checkbox to hide or show the plug-ins. A checkmark indicates that you want to show plug-ins that have the `bea.hidden` parameter set to `true`. A blank box (unchecked) indicates that you want to hide these “`bea.hidden`” logic plug-ins on the administration console. When set as desired, click **OK** to save the preference.

   All relevant screens on the C-Hub Administration Console will now reflect the new setting.

CHAPTER

18 Monitoring the C-Hub

The following sections describe how to monitor a c-hub:

- Monitoring C-Spaces on the C-Hub
- Monitoring Trading Partners on the C-Hub
- Monitoring Conversations on the C-Hub
- Viewing Messages on the C-Hub
- Viewing the WebLogic Collaborate Log

Viewing Statistics on the C-Hub

To view statistics on the current c-hub, select **Monitoring** in the left navigation bar, then click the **Hub** tab.

Figure 18-1 Viewing Statistics on the C-Hub

The following table explains the fields on this screen.

Table 18-1 Monitoring Trading Partners Fields

Field	Description
Active Since	Shows the date and time this c-hub was started.
Active C-Spaces	Shows the number of active c-spaces for this c-hub. (A c-space is active when it has at least one trading partner connected to it.)
Active Conversations	Shows the number of active conversations for this c-hub.
Active Trading Partners	Shows the number of active trading partners for this c-hub. (A trading partner is active when it is connected to at least one c-space.)

Table 18-1 Monitoring Trading Partners Fields

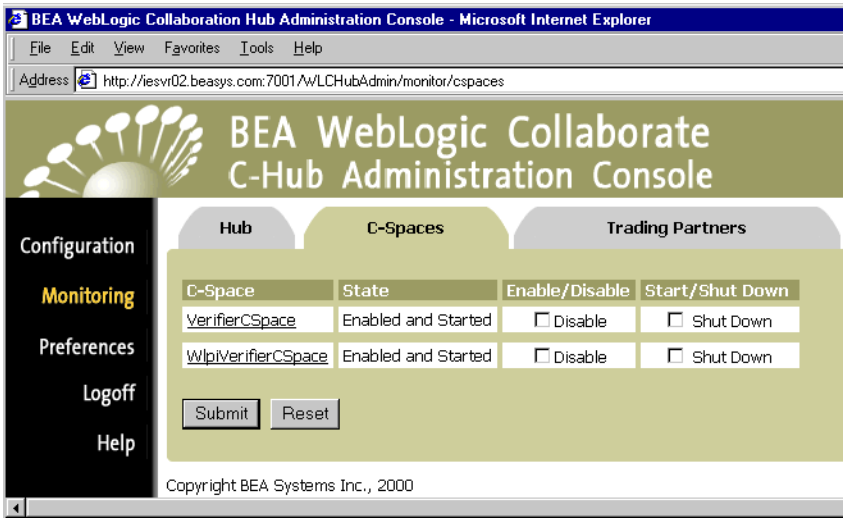
Field	Description
Total C-Spaces	Shows the total number of c-spaces for this c-hub.
Conversations Defined	Shows the number of conversations defined for this c-hub.
Messages Received	Shows the number of messages received in this c-hub.
Messages Sent	Shows the number of messages sent in this c-hub.
Last Received	Shows the date and time that the last message was received.
Last Sent	Shows the date and time that the last message was sent.

Note: You can also shut down a running c-hub from this tab. For more information on this, see "Shutting Down a Running C-Hub" on page 9-7.

Monitoring C-Spaces on the C-Hub

To view c-spaces for a c-hub, select **Monitoring** in the left navigation bar, then click the **C-Spaces** tab.

Figure 18-2 Monitoring C-Spaces



The following table explains the fields on this screen.

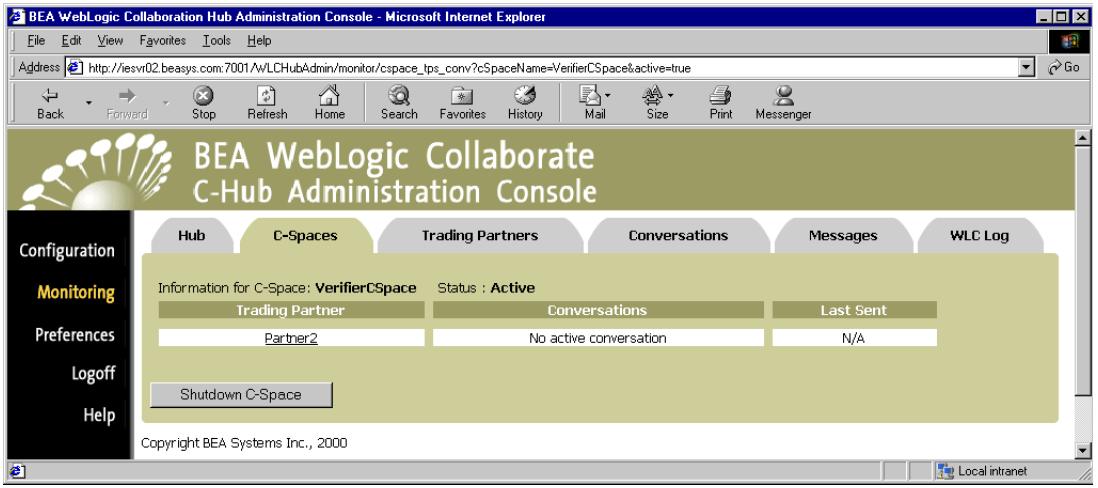
Table 18-2 Monitoring C-Spaces on a C-Hub - Main Screen

Field	Description
C-Space	Shows the names of the c-spaces configured for this c-hub. You can click on a c-space name to view details on the selected c-space. (See "Getting Details on a C-Space" on page 18-6.)
State	Shows the state of the c-space. The states are Enabled or Disabled and Started or Shut Down. Note that any combination of Enabled/Disabled and Started/Shut Down is possible, as explained in the following description of each state: ■ **Enabled**. The c-space is accepting new conversations (if started). ■ **Disabled**. The c-space is not accepting new conversations, but if the c-space is started (running) any existing conversations will be allowed to complete. ■ **Started**. The c-space is activated and running on the c-hub. If it is also enabled, it is accepting new conversations. If it is disabled, it is not excepting new conversations. ■ **Shut Down**. The c-space is not running. Any previously existing conversations were terminated.
Enable/Disable	To enable a c-space, select Enable and click Submit. To disable a c-space, select Disable and click Submit. ■ **Enable**. The c-space will accept new conversations (if started). ■ **Disable**. The c-space will not accept new conversations. Disabling an active (started) c-space before shutting it down allows for a graceful shutdown. Ideally, once you disable a c-space, you should see existing conversations complete and the c-space eventually become inactive (since no new conversations are allowed). If you wait until there is no conversation activity, you can shut down the c-space without disrupting any conversations.
Start/Shut Down	To shut down an active c-space, select the c-space in the Shut Down list and click **Submit**. To start a c-space, select the c-space in the Shut Down list and click **Submit**.

Getting Details on a C-Space

From the main C-Spaces tab you can click on a value in the C-Spaces field to view the details about the selected c-space.

Figure 18-3 Details on a Selected C-Space



The following table explains the fields on this screen.

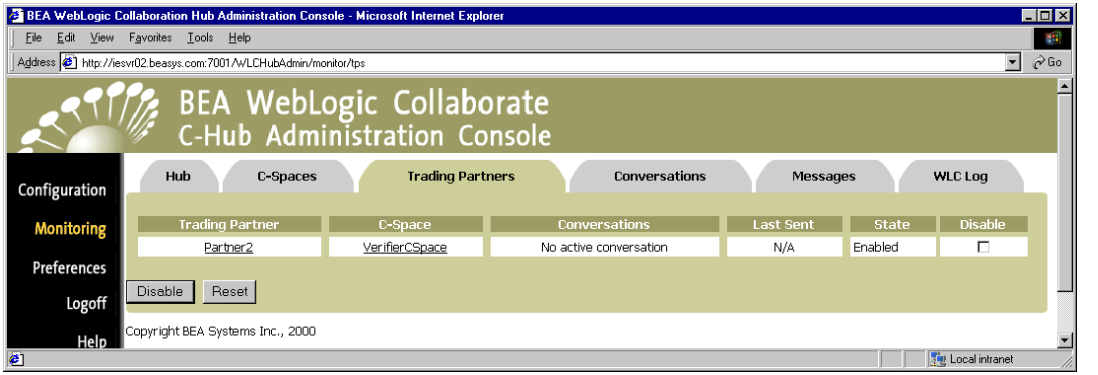
Table 18-3 Monitoring Details on a C-Space

Field	Description
Trading Partner	Shows the trading partners participating in this c-space.
Conversations	Shows the conversations in this c-space.
Last Sent	Shows the date and time of the last message this trading partner sent.

Monitoring Trading Partners on the C-Hub

To view trading partners for a c-hub, select **Monitoring** in the left navigation bar, then click the **Trading Partners** tab.

Figure 18-4 Monitoring Trading Partners



The following table explains the fields on this screen.

Table 18-4 Monitoring Trading Partners Fields

Field	Description
Trading Partner	Shows the trading partner name. You can click on a trading partner name to go to the trading partner detail page. (See "Getting Details on a Trading Partner" on page 18-8.) **Note:** If a trading partner is displayed in a red box, this indicates that the last message sent to this trading partner was unsuccessful (that is, the message did not reach the trading partner). If you notice that a trading partner is unavailable over a long period of time, you might want to force that trading partner to leave the c-space. Doing so can save the c-hub from continued failed attempts to send messages to an unavailable trading partner.
C-Space	Shows the name of the C-space this trading partner is in. You can click on this name to go to the C-Space Detail screen.
Conversations	Shows the conversations this trading partner is participating in.

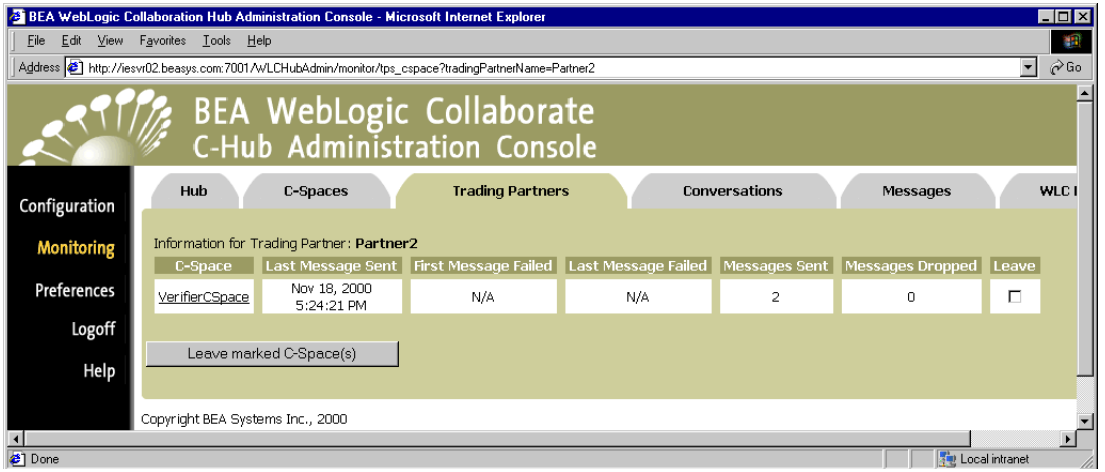
Table 18-4 Monitoring Trading Partners Fields

Field	Description
Last Sent	Shows the date and time of the last message this trading partner sent.
State	Shows whether the trading partner is enabled or disabled.
Disable	To disable a trading partner, select this checkbox and click the **Disable** button. A trading partner that is disabled cannot participate in new conversations, but can continue with existing conversations.

Getting Details on a Trading Partner

From the main Trading Partners monitoring tab you can click on a value in the Trading Partner field to view the details about the selected trading partner.

Figure 18-5 Details on a Selected Trading Partner



The following table explains the fields on this screen.

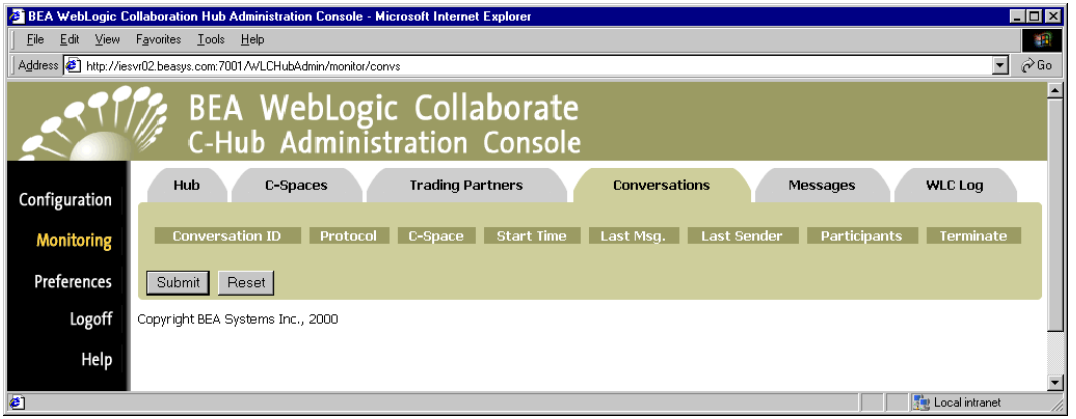
Table 18-5 Monitoring Trading Partners Fields

Field	Description
C-Space	Shows the name of the c-space this trading partner is in. You can click on this name to go to the c-space detail screen.
Last Message Sent	Shows the date and time of the last message sent by this trading partner in this c-space.
First Message Failed	Shows the date and time of the first failed message sent by this trading partner in this c-space.
Last Message Failed	Shows the date and time of the last failed message sent by this trading partner in this c-space.
Messages Sent	Shows the number of messages sent by this trading partner in this c-space.
Messages Dropped	Shows the number of messages dropped by this trading partner in this c-space.
Leave	Select Leave for c-spaces that you want this trading partner to leave. When you click the **Leave marked C-Space(s)** button, the trading partner will leave all c-spaces that are selected (checkmarked) in the Leave column. All active conversations for the trading partner in the selected c-space will end.

Monitoring Conversations on the C-Hub

To view conversations for a c-hub, select **Monitoring** in the left navigation bar, then click the **Conversations** tab.

Figure 18-6 Monitoring Conversations



The following table explains the fields on this screen.

Table 18-6 Monitoring Conversations Fields

Field	Description
Conversation ID	Shows the conversation ID.
Protocol	Shows the business protocol assigned for the conversation.
C-Space	Shows the name of the c-space this conversation is in. You can click on this name to go to the C-Space detail screen.
Start Time	Shows the date and time this conversation was started.
Last Msg.	Shows the date and time the last message was sent.
Last Sender	Shows the name of the trading partner who sent the last message.
Participants	Shows the names of the trading partners in this conversation.

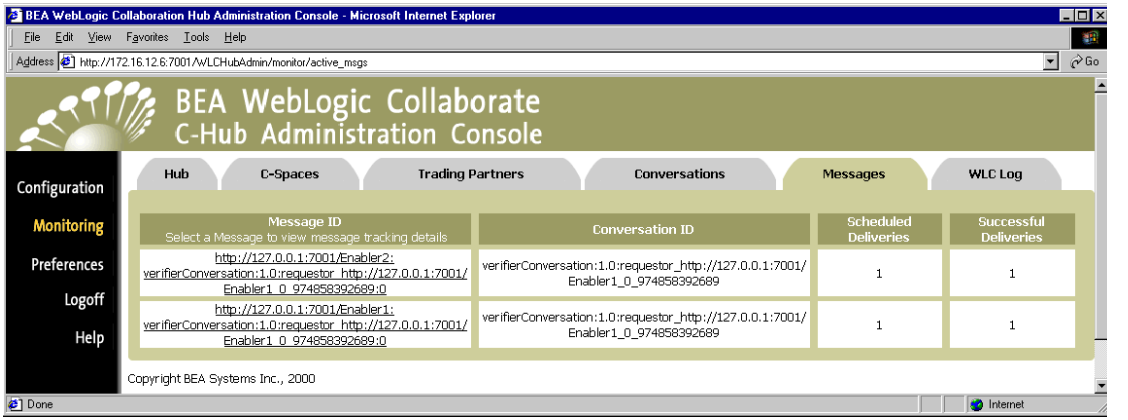
Table 18-6 Monitoring Conversations Fields

Field	Description
Terminate	You can *Terminate* any conversation on the c-hub with a choice of either *Success* or *Failure*. Check Success to indicate that the conversation completed successfully. Check Failure to indicate that there was an error in the conversation, and that is why you are terminating it.

Viewing Messages on the C-Hub

To view *outstanding* Messages for a c-hub, select **Monitoring** in the left navigation bar, then click the **Messages** tab. (Outstanding messages are messages that have not been delivered to all scheduled destinations.)

Figure 18-7 C-Hub Messages



The following table explains the fields on this screen.

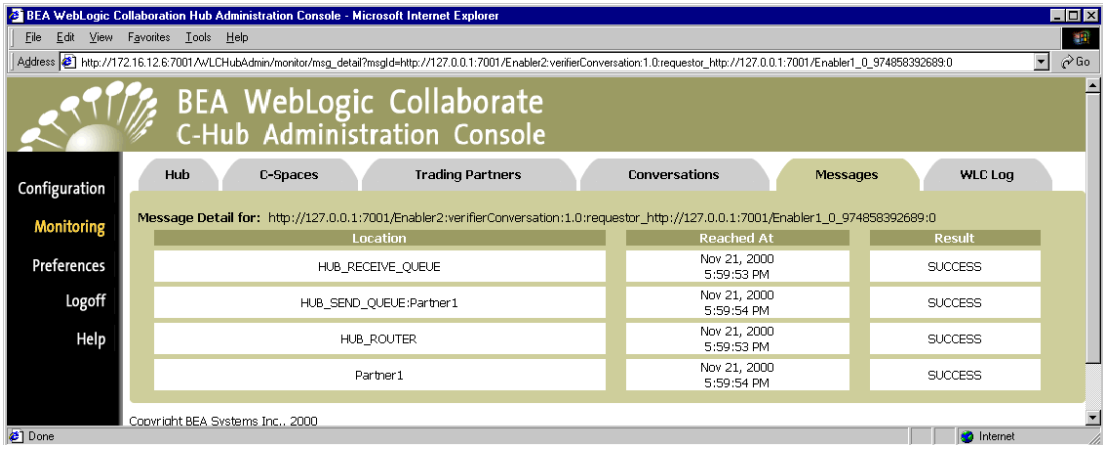
Table 18-7 Monitoring Messages Fields

Field	Description
Message ID	Message ID generated by WebLogic Collaborate.
Conversation ID	Conversation ID of the active conversation.
Scheduled Deliveries	Number of destinations to which this message is scheduled to be delivered.
Successful Deliveries	Number of destinations this message has already reached.

Viewing Details for a Particular Message

From the main Message monitoring tab you can click on a Message in the Message ID column to view the details about the selected message.

Figure 18-8 Viewing Details for the Selected Message



The following table explains the information reported on the details of a particular message.

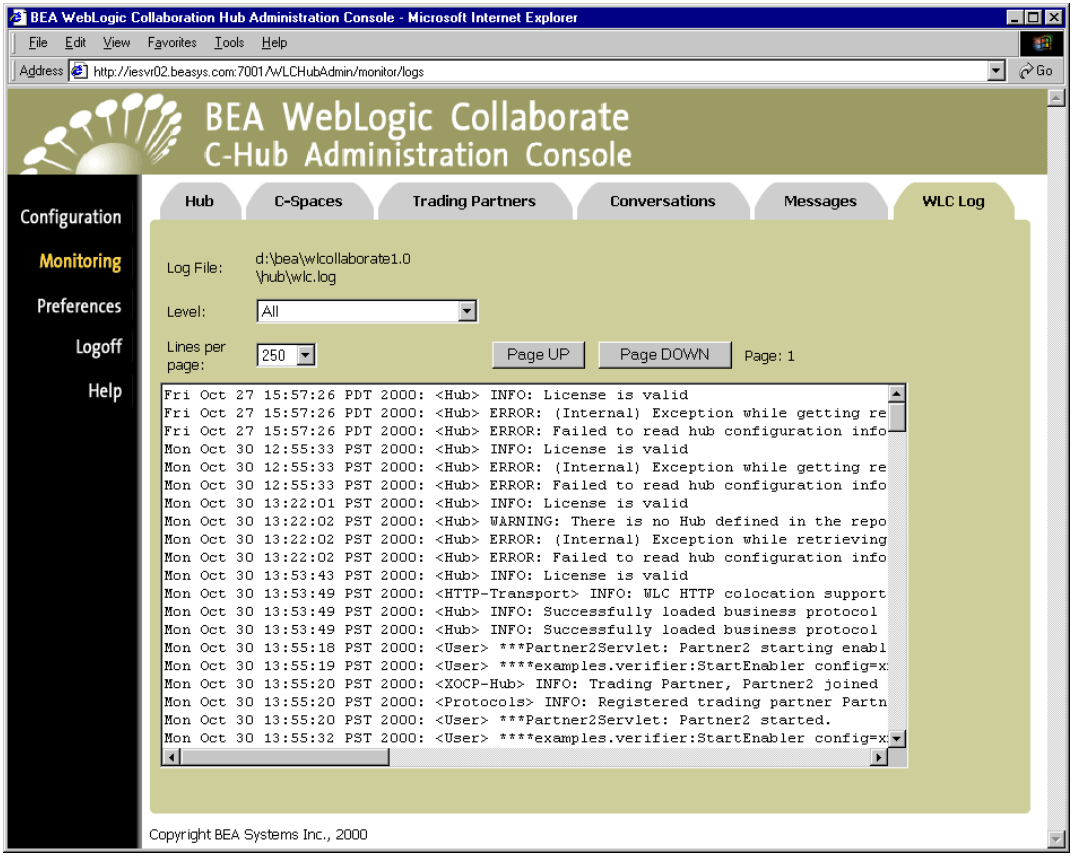
Table 18-8 Details on a Messages

Field	Description
Location	Locations to which the message has been sent.
Reached At	■ If the message was successfully sent, this field indicates the date and time the message arrived at the location. ■ If the message failed to arrive, this field indicates the last date and time WebLogic Collaborate *attempted* to send the message.
Result	The possible results are: ■ Success ■ Failure ■ Retries exhausted

Viewing the WebLogic Collaborate Log

To view WebLogic Collaborate (WLC) log, select **Monitoring** in the left navigation bar, then click the **WLC Log** tab.

Figure 18-9 Viewing the WLC Log



You can set the Level and Lines per Page filters in the drop-down menus as explained in the following table.

Table 18-9 Monitoring Trading Partners Fields

Field	Description
Log File	Shows the name and location of the log file. (The name is always `wlc.log` and it is always located in the same directory from where the WebLogic Server instance was started.)
Level	A drop-down menu where you can select the severity of the messages to be retrieved from the log. The choices are: ■ **All** – Displays all messages ■ **Info** – Displays any message with "INFO" or greater severity ■ **Warning** - Displays any message with "Warning" or greater severity ■ **Error** - Displays any message with "Error" or greater severity ■ **Fatal** - Displays any message with "Fatal" severity
Lines Per Page	A drop-down menu where you can select the number of messages to retrieve from the log. The choices are: ■ **250** ■ **500** ■ **1000** ■ **5000**

A list of error messages is dynamically displayed based on the Level and Lines per Page filters you apply. You can use the Page UP and Page DOWN buttons to browse through long lists of messages.

Index

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X