BEA WebLogic Collaborate

A Component of BEA WebLogic Integration

Using BEA WebLogic Collaborate Security

BEA WebLogic Collaborate Release 2.0
Document Edition 2.0
July 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Using BEA WebLogic Collaborate Security

Document Edition	Date	Software Version
2.0	July 2001	2.0

Contents

3. Configuring Security

4. Implementing Nonrepudiation

About This Document

This document describes how to implement a security scheme for your WebLogic Collaborate™ deployment.

This document is organized as follows:

- Chapter 1, “Introducing WebLogic Collaborate Security,” provides an overview of WebLogic Collaborate security and explains how it is based on WebLogic Server security.
- Chapter 2, “Authenticating and Authorizing Trading Partners,” describes the authentication and authorization processes used by the WebLogic Collaborate software.
- Chapter 3, “Configuring Security,” explains how to configure security for your WebLogic Collaborate trading partners and environment.
- Chapter 4, “Implementing Nonrepudiation,” explains how to implement a nonrepudication mechanism in your business processes.
- Chapter A, “Using the Secure Fingerprint Utility,” explains how to use the Secure Fingerprint Utility to extract the fingerprint value from a digital certificate.

What You Need to Know

This document is intended primarily for:

- Business analysts and programmers who design security mechanisms for their WebLogic Collaborate deployments

- System administrators who will set up and administer WebLogic Collaborate security.

For an overview of the WebLogic Collaborate architecture, see *Introducing BEA WebLogic Collaborate*.

How to Print this Document

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

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

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

Contact Us!

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

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

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

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes

- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

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

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

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Introducing WebLogic Collaborate Security

This topic includes the following sections:

- WebLogic Collaborate Security Model
- Principals, Users, and Groups
- Digital Certificates
- Certificate Authority
- SSL Protocol
- Configuration Restrictions to Ensure a Secure Environment

WebLogic Collaborate Security Model

The WebLogic Collaborate security model incorporates the following primary features:

- Uses the security features of the underlying BEA WebLogic Server™ platform to perform authentication and authorization of principals before granting access to WebLogic Collaborate resources.
- Is extensible by allowing you to incorporate your own or third-party vendor tools to verify trading partner digital certificates and implement nonrepudiation support, which is a requirement for critical business messages.

This section describes the WebLogic Server and WebLogic Collaborate entities involved in providing the authentication and authorization features of WebLogic Collaborate.

WebLogic Collaborate *authentication* is the process of verifying a principal's identity. Authentication is concerned with who an entity is; it is the association of an identity with an entity. Authorization is concerned with what that identity is allowed to see and do. WebLogic Collaborate uses the following methods to perform authentication:

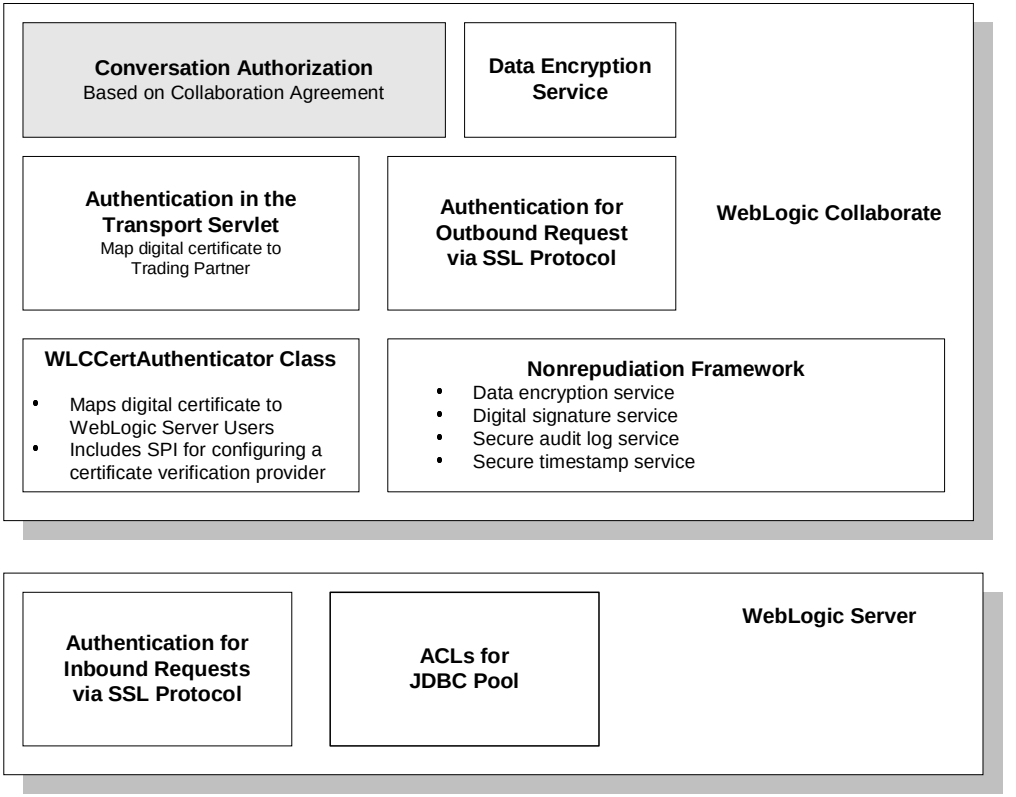
- Username and password—human users (administrators) use usernames and passwords to prove their identity.
- Digital certificates—trading partners in WebLogic Collaborate use digital certificates to prove their identity to WebLogic Collaborate.
- Secure sockets layer (SSL)—the SSL protocol provides data integrity and confidentiality to the connections between principals.

Authorization protects who has access to the available resources. Permission to access WebLogic Collaborate resources is assigned through access control lists (ACLs) and roles.

For complete details about how WebLogic Server and WebLogic Collaborate work together to authenticate and authorize principals in WebLogic Collaborate, see Chapter 2, "Authenticating and Authorizing Trading Partners."

The following figure shows the entities and features in WebLogic Server and WebLogic Collaborate that provide the WebLogic Collaborate security model.

Figure 1-1 WebLogic Collaborate Security Model



The following table describes each of the features shown in this WebLogic Collaborate security model.

Table 1-1 Components in the WebLogic Collaborate Security Model

Component	Description
Conversation authorization	When a trading partner business message arrives, WebLogic Collaborate, as part of the business message authorization process, examines the contents of the business message to validate it against the collaboration agreement. That is, the collaboration agreement defines the business messages a given trading partner may send and receive. WebLogic Collaborate verifies that the content of the incoming business message is consistent with the business messages that the trading partner is bound, by role and conversation definition in the collaboration agreement, to either send or receive. This authorization scheme makes sure that only the business messages that are consistent with the relevant collaboration agreement have access to WebLogic Collaborate resources.
Data encryption service	The data encryption service encrypts business messages for the business protocols that require it. Data encryption works by using a combination of the sender's certificate, private key, and the recipient's certificate to encode the business message. The message can then be decrypted only by the recipient using the recipient's private key. For details about using the data encryption service, see "Configuring Message Encryption" on page 3-29.
Authentication in the transport servlet	A transport servlet is a WebLogic Collaborate-specific servlet that serves as the entry point for both HTTP and HTTPS access to WebLogic Collaborate resources, including the following: ■ WebLogic Collaborate repository ■ WebLogic Process Integrator workflow templates and definitions ■ JDBC connection pool A transport servlet is dynamically registered in the WebLogic Server environment for trading partners bound to a specific collaboration agreement.

Table 1-1 Components in the WebLogic Collaborate Security Model

Component	Description
Authentication for outbound request via the SSL protocol	WebLogic Collaborate authenticates the recipient for all outbound messages using the SSL certificate obtained in SSL handshake to ensure that the messages are consistent with the relevant collaboration agreement to which they are bound.
`WLCCertAuthenticator` class	The `WLCCertAuthenticator` class maps trading partner certificates to the corresponding WebLogic Server users that have been configured for the trading partner. The `WLCCertAuthenticator` class implements the `weblogic.security.acl.CertAuthenticator` interface. You can configure this class to invoke your own or a trusted third-party vendor's implementation that verifies trading partner certificates. For more information, see Chapter 2, "Authenticating and Authorizing Trading Partners."
Nonrepudiation framework	The WebLogic Collaborate security system provides a means to implement nonrepudiation support. Nonrepudiation is the ability of a trading partner to prove or disprove having previously sent or received a particular business message to or from another trading partner. Nonrepudiation requires the following services: ■ Data encryption ■ Digital signatures ■ Secure timestamps ■ Secure audit log WebLogic Collaborate provides out-of-the-box implementations for nonrepudiation and Service Provider Interfaces (SPIs) that allow you to incorporate your own or a trusted third-party's implementation. For more information about nonrepudiation, see Chapter 4, "Implementing Nonrepudiation."

Table 1-1 Components in the WebLogic Collaborate Security Model

Component	Description
Authentication for inbound requests via SSL protocol	When an inbound trading partner message arrives, both the trading partner and the WebLogic Server system exchange certificates to establish each other's identity. When the SSL handshake is completed, the trading partner's network connection to the WebLogic Server system is established. For information about configuring the SSL protocol in WebLogic Server to provide mutual authentication, see "Configuring the SSL Protocol and Mutual Authentication" on page 3-2.
ACLs for JDBC connection pool	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 specific 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. For information about configuring the ACLs for the JDBC connection pool, see "Configuring Access Control Lists for WebLogic Collaborate" on page 3-6.

For more 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*.

Principals, Users, and Groups

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

- Trading partners
- Human users—WebLogic Collaborate administrators

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

WebLogic Collaborate supports the following types of WebLogic Server users:

- Trading partner users on WebLogic Collaborate
- WebLogic Collaborate system user
- WebLogic Collaborate administrator

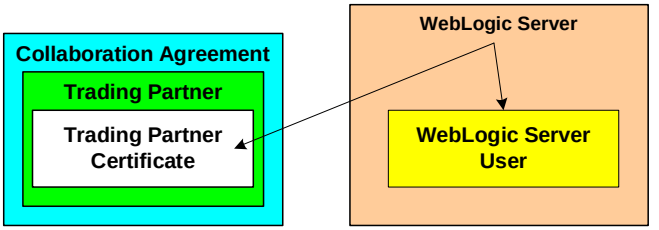
Groups are sets of WebLogic Server users. Groups provide an efficient way to manage large numbers of users because an administrator can specify permissions for an entire group at one time.

Note: Because the tasks typically performed via the WebLogic Collaborate Administration Console requires system level privileges—for creating users, accessing MBeans, and so on—the WebLogic Server username for logging into the WebLogic Collaborate Administration Console is configured as a WebLogic Server system user.

About Configuring Trading Partners

When you configure a collaboration agreement in WebLogic Collaborate, you also specify the trading partner name bound to that agreement. To associate a user with a trading partner in the WebLogic Collaborate Administration Console, specify the trading partner username, which is a WebLogic Server username. WebLogic Server maps the digital certificate for that trading partner to the trading partner user at run time.

Figure 1-2 Mapping a Trading Partner Certificate to a WebLogic Server User



Therefore, when a trading partner message arrives in WebLogic Server, WebLogic Server is able to match a trading partner to a WebLogic Server user by reading a trading partner certificate, and the WebLogic Collaborate authentication process may begin.

About Configuring the WebLogic Collaborate System User

Please note the following about the WebLogic Collaborate system user:

- The WebLogic Collaborate system user has access to all WebLogic Collaborate resources except the transport servlet. This restriction prevents an external entity from entering the WebLogic Collaborate system as a WebLogic Collaborate system user.
- The WebLogic Collaborate system user is predefined in the sample configuration shipped with the product. The default password for `wlcsystem` user is `wlcsystem`. However, if the system user does not exist for any reason, you can

create the username `wlcsystem` (password `wlcsystem`) via the WebLogic Server Administration Console. (You may also change the password as desired in the WebLogic Collaborate Administration Console.)

- Do not use the WebLogic Server Administration Console to modify a WebLogic Collaborate system password. The password is stored in the repository for run-time access to WebLogic Collaborate resources.

  Note: When the password does not match the one specified in the `fileRealm.properties` file, a warning is entered in the system log. When the password for user `wlcsystem` does not match, WebLogic Collaborate uses the user `system` for run-time access to the repository.

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 certificate authority's 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 certificate authority.

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 e-mail 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)

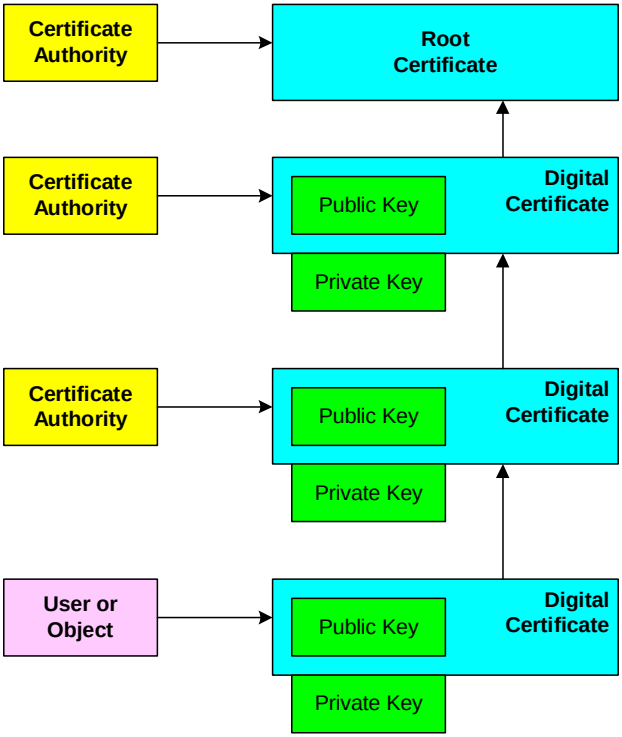
The most widely accepted format for digital certificates is defined by the ITU-T X.509 international standard. Thus, digital certificates can be read or written by any application complying with the X.509 standard. The public key infrastructure (PKI) in WebLogic Server recognizes digital certificates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

Figure 1-3 Certificate Authority Hierarchy



Before you use a digital certificate, verify a digital signature, or decrypt a business message, make sure that the digital certificate is issued by a trusted certificate authority. Regardless of who encrypts the business message, the digital certificate of the business message must be trusted, which is established by the certificate authority.

SSL Protocol

The SSL protocol provides secure connections by enabling two applications linked through 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 trading partners.

Administrators use a Web browser to access the WebLogic Collaborate Administration Console. You can use the Hypertext Transfer Protocol with SSL (HTTPS) to secure this type of network communication.

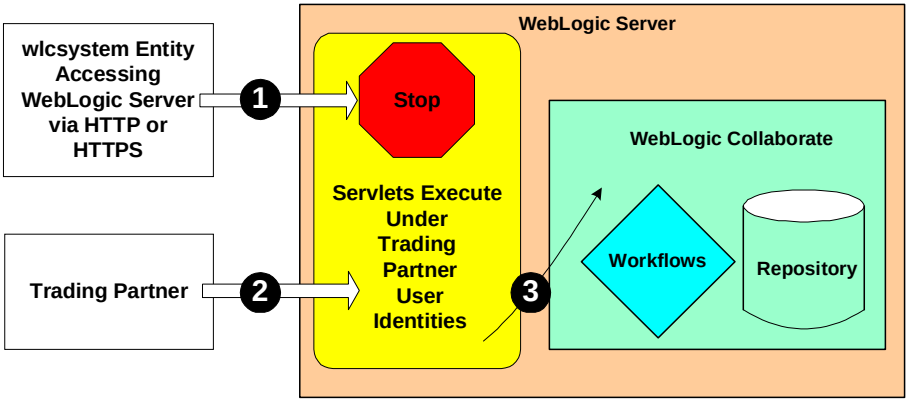
Configuration Restrictions to Ensure a Secure Environment

WebLogic Collaborate imposes the restrictions described in this section to ensure a secure environment. Some of these restrictions are repeated, as appropriate, in Chapter 3, "Configuring Security."

- The WebLogic Collaborate system user is not authorized to access the transport servlet. This ensures that no external entity can impersonate the WebLogic Collaborate system user.
- Trading partners are not authorized to access WebLogic Collaborate resources. (After a trading partner certificate has been authenticated, the trading partner certificate is mapped to a WebLogic Server user. Only after the trading partner business message has also been authenticated, the WebLogic Server user to whom the trading partner certificate has been mapped accesses WebLogic Collaborate resources on the trading partner's behalf.)
- The architecture of the WebLogic Collaborate environment is designed so that there is never a need to divulge password information for trading partners because trading partners are always mapped in the WebLogic Collaborate environment from their digital certificates.

The following figure shows how these security restrictions appear in the WebLogic Collaborate security model.

Figure 1-4 The Secure WebLogic Collaborate Environment



In the preceding figure, note the following callouts:

1. Any entity named `wlcsystem` attempting to gain access to the WebLogic Collaborate transport servlet is denied access.
2. After the trading partner certificate and business message are validated, the trading partner certificate is mapped to the corresponding WebLogic Server user.
3. The WebLogic Server user mapped in previous step accesses the WebLogic Collaborate resources required to service the trading partner business message.

CHAPTER

2 Authenticating and Authorizing Trading Partners

The topic includes the following sections:

- Trading Partner Authentication in WebLogic Collaborate
- Trading Partner Authorization in WebLogic Collaborate

Trading Partner Authentication in WebLogic Collaborate

Authentication is the process by which WebLogic Collaborate establishes the identity of a principal. Digital certificates using the SSL protocol with mutual authentication (HTTPS) are used between a trading partner and WebLogic Collaborate. WebLogic Collaborate examines and validates digital certificates against security information stored in the repository.

WebLogic Collaborate incorporates a two-level authentication process:

- The first level involves verification of the trading partner certificate.
- The second level involves authentication of the trading partner message.

When a trading partner business message has passed both levels of authentication, WebLogic Collaborate performs the authorization process on the business message.

The sections that follow describe both levels of the WebLogic Collaborate authentication process.

Trading Partner Certificate Verification

The WebLogic Collaborate security model provides a Service Provider Interface (SPI) that allows you to insert a Java class that implements an interface that calls out to a third-party service to verify trading partner certificates. Such an implementation, called a certificate verification provider (CVP), can call out to one of the following certificate verification applications:

- A Certificate Revocation List (CRL) implementation
- An Online Certificate Status Protocol (OCSP) implementation that interacts with a trusted third-party entity, such as a certificate authority, for real-time certificate status checking
- Your own certificate verification implementation

Benefits of Certificate Verification

The purpose of trading partner certificate verification is to validate the trading partner's digital certificate. For example, verifying a certificate may involve some or all of the following tasks:

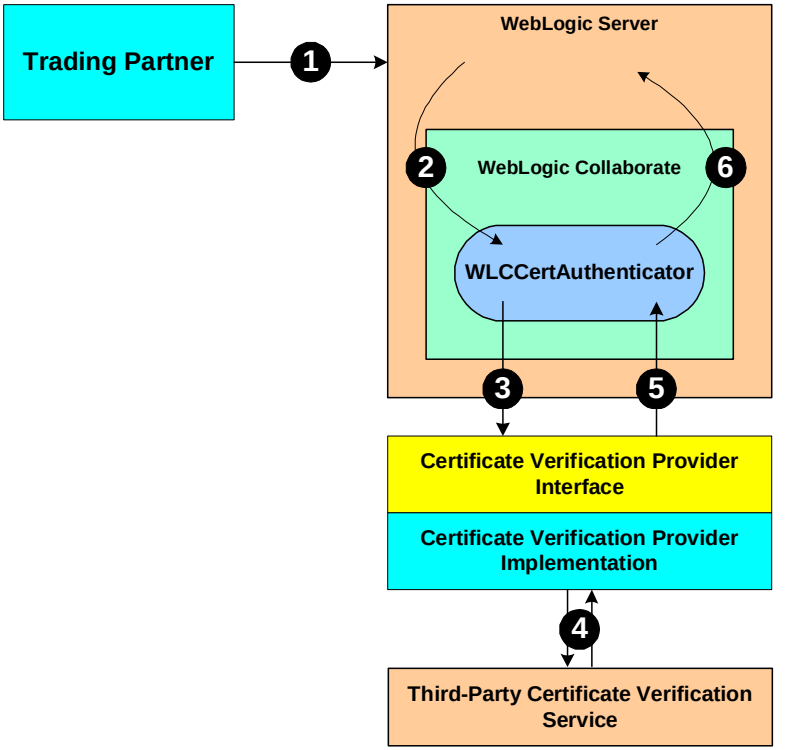
- Traversing the certificate chain to the root certificate authority
- Checking a certificate revocation list (CRL) for all the certificates in the chain to identify any of those that have been revoked
- Performing a real-time certificate check with a trusted vendor, who can verify the certificate
- Checking to make sure all dates in the certificate chain are valid
- Verifying the signature of each certificate in the chain

Configuring and using a CVP implementation is optional, but doing so can provide an additional level of security in the certificate verification process.

Certificate Verification Process

The following figure shows the sequence of events that occur during the certificate verification process in the WebLogic Collaborate environment.

Figure 2-1 Trading Partner Certificate Verification in WebLogic Collaborate



In the preceding figure, note the following callouts.

Callout	Description
1	Certificate verification is used only in SSL. The trading partner and the WebLogic Server system perform an SSL handshake, during which they exchange certificates to establish each other's identity. The Certificate Authority of the trading partner digital certificate must be trusted in WebLogic Server. During this handshake, WebLogic Server verifies the following: ■ The Certificate Authority of the trading partner certificate must be one that is trusted in the WebLogic Server environment. ■ The trading partner certificate has not expired. When the SSL handshake is completed, the trading partner's network connection to the WebLogic Server system is established.
2	WebLogic Server invokes the `WLCCertAuthenticator` class in WebLogic Collaborate. The `WLCCertAuthenticator` class in turn implements the `weblogic.security.acl.CertAuthenticator` interface in order to map the trading partner certificate to the corresponding WebLogic Server user that has been configured for the trading partner.
3	The `WLCCertAuthenticator` class invokes the CVP interface to the implementation that calls out to the third-party certificate verification service.
4	The CVP implementation calls out to the third-party certificate verification service, which returns the status of the trading partner certificate.
5	The CVP implementation returns the appropriate status of the certificate to the `WLCCertAuthenticator` class.
6	If the trading partner certificate is valid, WebLogic Collaborate attempts to map the certificate to a valid trading partner name in the repository. If the certificate maps to a valid trading partner, WebLogic Collaborate returns a WebLogic Server user to WebLogic Server.

Implementing a Certificate Verification Provider

A certificate verification provider (CVP) Java class must implement the `com.bea.b2b.security.CertificateVerificationProvider` interface. You have three choices for what a CVP class can call out to:

- A trusted third-party vendor that conforms to the service provider interface, as described in "Using the Service Provider Interface" on page 2-5.
- The BEA-provided CVP application that you can download from the Developer Center, available at the following URL:

  `http://developer.bea.com/index.jsp`
- Your own certificate verification application.

Regardless of which choice you pick, you need to create a Java implementation of the CVP SPI that calls out to the application that performs the actual certificate verification. Creating, compiling, and configuring this CVP application is explained in the subsections that follow.

Using the Service Provider Interface

WebLogic Collaborate allows you to implement a CVP via the `com.bea.b2b.security.CertificateVerificationProvider` interface, which provides the CVP service provider interface (SPI). If you implement or use a CVP using the SPI described in this section, you must later configure this CVP in the WebLogic Collaborate Administration Console so that the CVP is invoked properly during run time.

The `com.bea.b2b.security.CertificateVerificationProvider` interface has the following methods, which a CVP application must implement:

- `void init()`

  This method is automatically invoked by WebLogic Collaborate to invoke any custom initialization processes in the class you create that implements this interface. This method is invoked only once, at the startup of WebLogic Collaborate.
- `String verify(Certificate[] certs)`

  This method validates the certificate chain obtained during the SSL handshake. It returns one of the following `String` values:

  - `good`—the trading partner certificate is valid and not expired.
  - `revoked`—the trading partner certificate has been revoked by one of the certificate authorities in the certificate chain, or the trading partner certificate has expired.

- unknown—none of the certificate authorities in the certificate chain is able to establish the validity of the trading partner certificate.

The implementer can choose the validation procedure performed by this method. For example, this method can check certificate revocation lists (CRLs) stored in files, it can check the certificate status in real-time using the Online Certificate Status Protocol (OCSP), or it can use any other mechanism, as appropriate.

Notes: If you implement a CVP, you need to add a default public constructor for the CVP with no arguments. Neither the constructor nor any methods in the class should throw any exceptions.

If you do not configure a CVP, any certificate issued by a trusted certificate authority is considered by the WebLogic Collaborate Server to be valid.

Compiling the Certificate Verification Provider Class

If you implement a CVP, note the following:

- After you create the CVP Java class, you must compile it and place it in the system CLASSPATH.
- You must configure the CVP via the WebLogic Collaborate Administration Console or the Bulk Loader utility. After you configure the CVP, restart WebLogic Server so that the CVP can take effect. If you do not configure a CVP, any certificate issued by a trusted certificate authority is considered by the WebLogic Collaborate Server to be valid.

Configuring a Certificate Verification Provider with WebLogic Collaborate

For complete details about using the WebLogic Collaborate Administration Console to configure a CVP, see “Configuring a Certificate Verification Provider Interface” on page 3-37. After you configure a CVP, restart WebLogic Server so that the CVP can take effect.

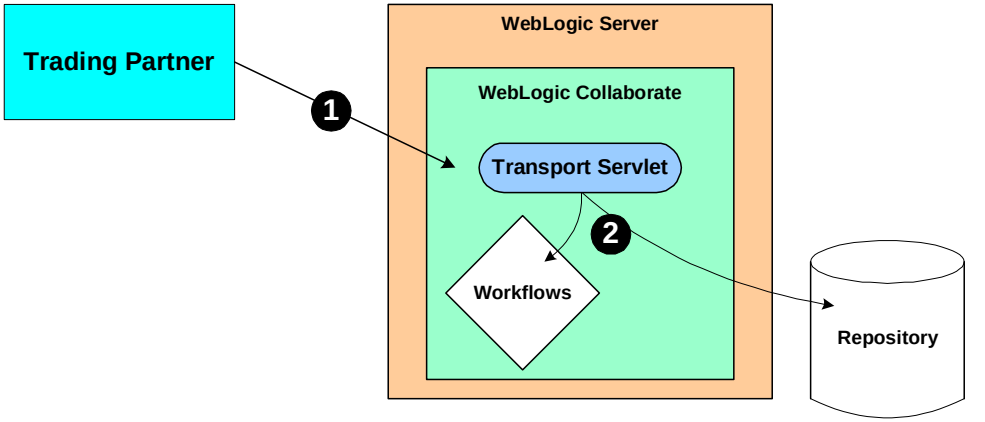
Authentication of the Trading Partner Message

After a trading partner’s certificate has been validated by WebLogic Server, the WebLogic Collaborate Server needs to authenticate the trading partner message before the message itself can be serviced. Authenticating the trading partner message involves

verifying that the sender of the business message is a valid trading partner listed in the WebLogic Collaborate repository. After a trading partner message has been authenticated, the trading partner's identity becomes recognized for full access to WebLogic Collaborate resources.

The following figure shows the process of authenticating a trading partner message.

Figure 2-2 Authenticating the Trading Partner Message



In the preceding figure, note the following:

- The transport servlet is the entry point into WebLogic Collaborate. When the trading partner message arrives in the WebLogic Collaborate transport servlet, as shown by callout 1, the transport servlet verifies the trading partner message. Verifying a trading partner means ensuring that the trading partner name is valid by retrieving its value from a valid certificate associated with the trading partner.
- When the trading partner message is authenticated, the trading partner is authorized for access to WebLogic Collaborate resources, such as the repository and WebLogic Process Integrator templates and workflows, shown by callout 2. The access is made available via the WebLogic Collaborate system user context.

Note: Only trading partners can be authenticated to use the WebLogic Collaborate transport servlet. If the WebLogic Collaborate system user attempts to access the transport servlet to access WebLogic Collaborate resources, the access is

denied by WebLogic Server. This mechanism ensures that no remote entity can gain access to WebLogic Collaborate resources assuming the identity of a WebLogic Collaborate system user.

Trading Partner Authorization in WebLogic Collaborate

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

The WebLogic Collaborate system incorporates two levels of authorization:

- Authorization of the trading partner for access to the WebLogic Collaborate transport servlet
- Authorization of the conversation associated with the trading partner business message

Trading Partner Authorization

This level of authorization is performed by WebLogic Server. When the trading partner message arrives in WebLogic Server, and the trading partner and WebLogic Server complete the mutual authentication procedure, the trading partner becomes authorized to access the WebLogic Collaborate transport servlet.

The path of the transport servlet is dynamic, so you need to edit the `web.xml` file to allow trading partners to access the URL of the transport servlet. You cannot preconfigure this because of the dynamic nature of the URL corresponding to the transport servlet in the WebLogic Collaborate environment.

You need to specify transport servlet ACLs in the `web.xml` file. The following example shows a `web.xml` file that specifies the ACLs for a transport servlet named `wlctransport`.

Listing 2-1 Example Transport Servlet ACL

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 1.2//EN
" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
...
...
<!-- Authentication -->
 <security-constraint>
  <web-resource-collection>
   <web-resource-name>wlctransport</web-resource-name>
   <url-pattern>*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
   <role-name>TradingPartnerGroupA</role-name>
  </auth-constraint>
 </security-constraint>

 <login-config>
  <auth-method>CLIENT_CERT</auth-method>
 </login-config>

 <security-role>
  <role-name>TradingPartnerGroupA</role-name>
 </security-role>
</web-app>
```

In the preceding code example:

- `wlctransport` is the transport servlet whose endpoint is defined in the WebLogic Collaborate repository.
- `TradingPartnerGroupA` is a WebLogic Server user group in which all the trading partner WebLogic Server users are members.
- `CLIENT_CERT` specifies that the mode of authentication required to access the transport servlet is SSL with mutual authentication.

Conversation Authorization

When WebLogic Collaborate performs conversation authorization, the server examines the content of the trading partner business message with respect to the collaboration agreement to which the trading partner is bound. That is, for a given role and party specified in a collaboration agreement, a trading partner may send only a specific set of business messages.WebLogic Collaborate validates the business message against the following information specified in the collaboration agreement for a particular conversation:

- Party information (trading partner and role)
- Conversation definition
- Document exchange ID

Once the conversation authorization is complete for an incoming business message, access to the WebLogic Collaborate resources is dictated by ACLs.

CHAPTER

3 Configuring Security

This topic includes the following sections:

- Configuring the SSL Protocol and Mutual Authentication
- Configuring Access Control Lists for WebLogic Collaborate
- Configuring Security for the WebLogic Collaborate System
- Configuring Trading Partner Security
- Configuring Message Encryption
- Configuring Digital Signatures for Nonrepudiation
- Customizing the WLCCertAuthenticator Class
- Configuring a Certificate Verification Provider Interface
- Configuring WebLogic Collaborate to Use an Outbound HTTP Proxy Server
- Configuring WebLogic Collaborate with a Webserver and a WebLogic Proxy Plug-In
- Configuring WebLogic Process Integrator Access to the WebLogic Collaborate Repository

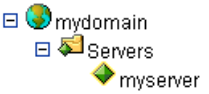
For general information about configuring WebLogic Collaborate, see "Configuration Tasks" in *Administering BEA WebLogic Collaborate*.

Configuring the SSL Protocol and Mutual Authentication

To configure WebLogic Server to use the SSL protocol and mutual authentication, complete the following steps:

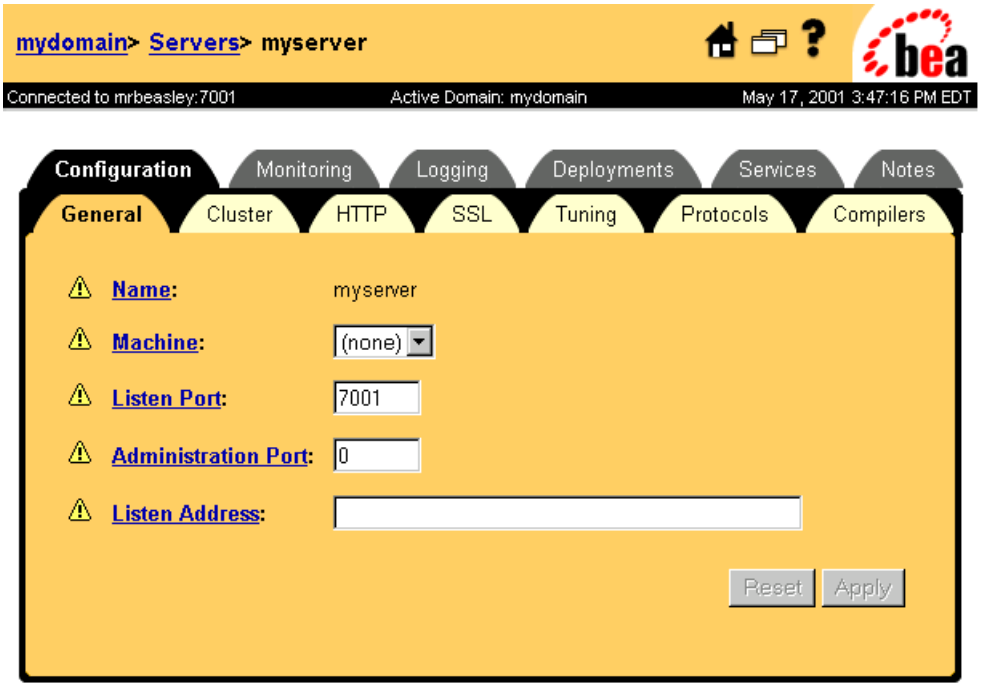
1. Obtain a digital certificate for WebLogic Server as described in "Configuring the SSL Protocol" in "Managing Security" in the *BEA WebLogic Server Administration Guide*.
2. Start the WebLogic Server Administration Console as described in "Starting the WebLogic Server Administration Console" in "Starting, Stopping, and Customizing WebLogic Collaborate" in *Administering BEA WebLogic Collaborate*.
3. In the navigation tree (in the left pane) of the WebLogic Server Administration Console, choose `Servers→myserver` for the domain you are configuring, as in the following figure.

Figure 3-1 Choosing a Domain



The Configuration page for WebLogic Server is displayed, shown in the following figure.

Figure 3-2 WebLogic Server Administration Console Configuration Page



4. Select the SSL tab to display the Secure Sockets Layer (SSL) configuration page, shown in the following figure.

Figure 3-3 SSL Configuration Page

5. The following table describes the information that you enter into the SSL configuration page.

Table 3-1 SSL Configuration Page Fields

Field	Description
Enabled check box	If checked, SSL connections are enabled between WebLogic Collaborate and trading partners.
SSL Listen Port	Specifies the dedicated port on which WebLogic Collaborate listens for SSL connections.

Table 3-1 SSL Configuration Page Fields (Continued)

Field	Description
Server Key File Name	Specifies the location of the private key file for WebLogic Server.
Server Certificate File Name	Specifies the location of the public key file for WebLogic Server. You obtain this file from a trusted security vendor, as described in step 1 in this section.
Server Certificate Chain File Name	Specifies the full directory location of the digital certificate for WebLogic Server. This location is also known as the root certificate authority.
Client Certificate Enforced check box	If checked, mutual authentication is enabled between WebLogic Collaborate and trading partners accessing WebLogic Collaborate resources.
Trusted CAFile Name	Specifies the name of the file that contains the digital certificate for the certificate authority trusted by WebLogic Server. Trading partners are required to present digital certificates issued by this certificate authority. You obtain this filename from each trading partner configured in your WebLogic Collaborate environment.
Certificate Authenticator	Specifies the certificate authenticator to be used to determine the validity of the trading partner digital certificate

Configuring Access Control Lists for WebLogic Collaborate

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 do the following:

1. Create an ACL for a resource.
2. Specify the permission for the resource.
3. Grant the permission to a specified set of users and groups.

For a WebLogic Collaborate resource, one or more permissions can be granted.

The ACL on the JDBC connection pool that is preset in the sample configuration shipped with WebLogic Collaborate has the following permissions set for the user `wlcsystem`: `reserve`, `reset`, and `shrink`.

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

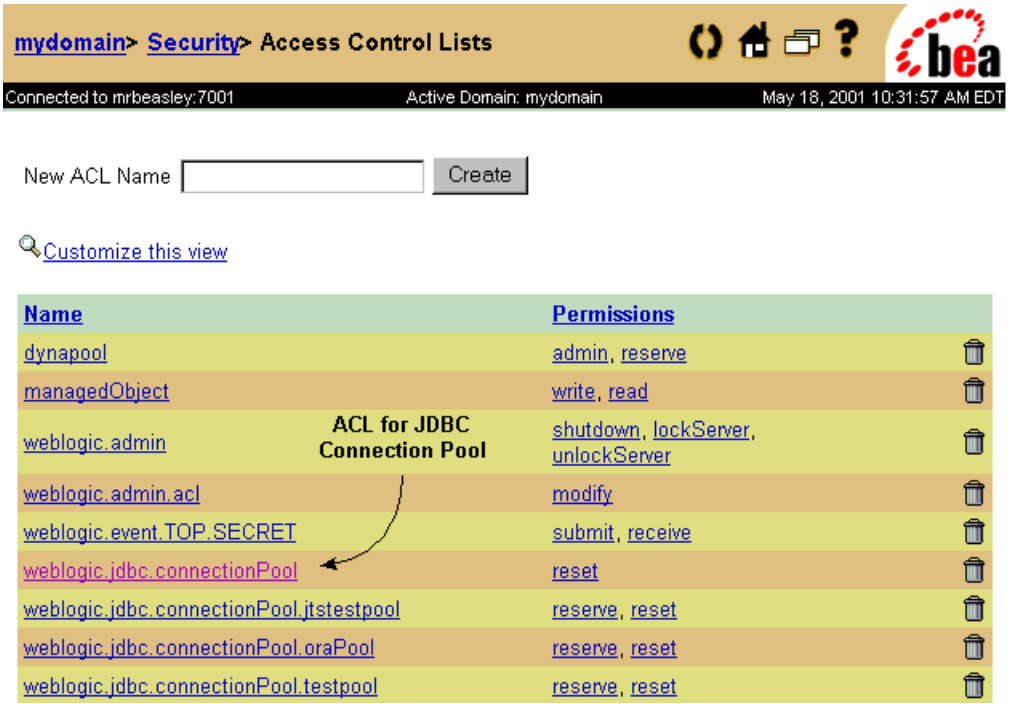
To set the ACLs on the JDBC connection pool:

1. Start the WebLogic Server Administration Console, if it is not already running.
2. In the navigation tree, choose Security→ACLs.

Figure 3-4 Choosing ACLs in the Navigation Tree

The ACLs that are configured in WebLogic Server are listed in the Access Control Lists configuration page, as shown in the following figure. Note the entry for the ACL for the JDBC connection pool.

Figure 3-5 ACL for the JDBC Connection Pool



3. Click the name of the ACL for the JDBC connection pool. The WebLogic Server Administration Console displays the dialog box in which you can set the required permissions for the JDBC connection pool, as shown in the following figure.

Figure 3-6 Setting Permissions for the JDBC Connection Pool

4. Click reset. The dialog box in which you can reset the ACLs for the JDBC connection pool is displayed, as shown in the following figure.

Figure 3-7 ACL Reset Dialog Box

5. Enter `wlcsystem` in the Users field, if necessary.
6. Click Grant Permission, if you have made any changes.

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

Configuring Security for the WebLogic Collaborate System

The WebLogic Collaborate repository contains security information about the WebLogic Collaborate system and the trading partners that access WebLogic Collaborate resources. You can configure repository information either by using the WebLogic Collaborate Administration Console, or by specifying it in a repository data file that you then import into the repository using the Bulk Loader.

Note: If you use the Bulk Migrator utility to migrate the repository from a previous release of WebLogic Collaborate, make sure the user `wlcsystem` is created and the correct password is included in the Bulk Loader data file. For more information about using the Bulk Migrator utility, see "Migrating the Repository" in *Migrating BEA WebLogic Collaborate to Release 2.0*.

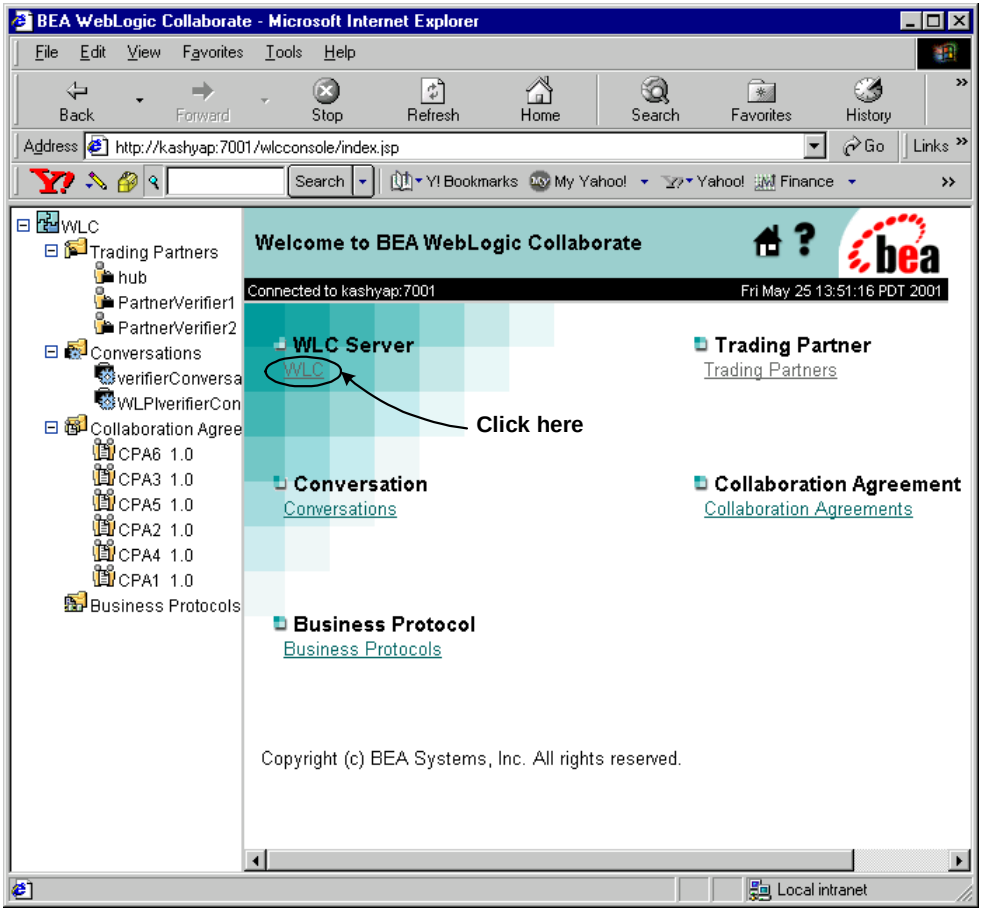
For the WebLogic Collaborate system, you need to configure the following as required:

- WebLogic Collaborate system password
- Audit log class
- Certificate verification class
- Secure timestamp class
- Certificate authority directory

To configure these entities in the WebLogic Collaborate system, complete the following steps:

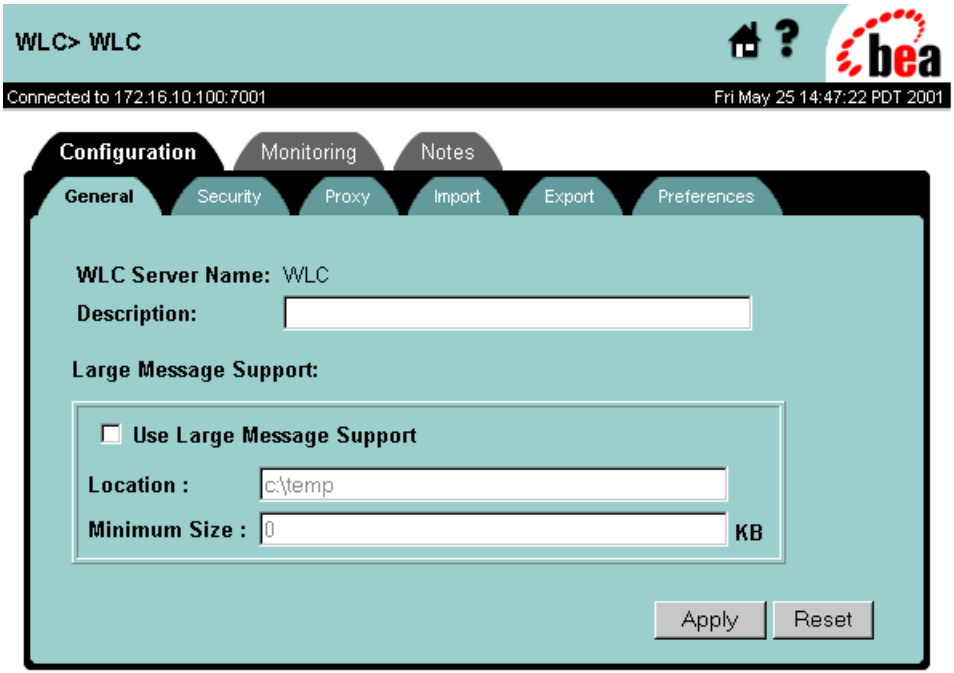
1. Start the WebLogic Collaborate Administration Console.
2. In the main pane of the WebLogic Collaborate Administration Console, click the link under WLC Server, as shown in the following figure.

Figure 3-8 WebLogic Collaborate Administration Console Main Window



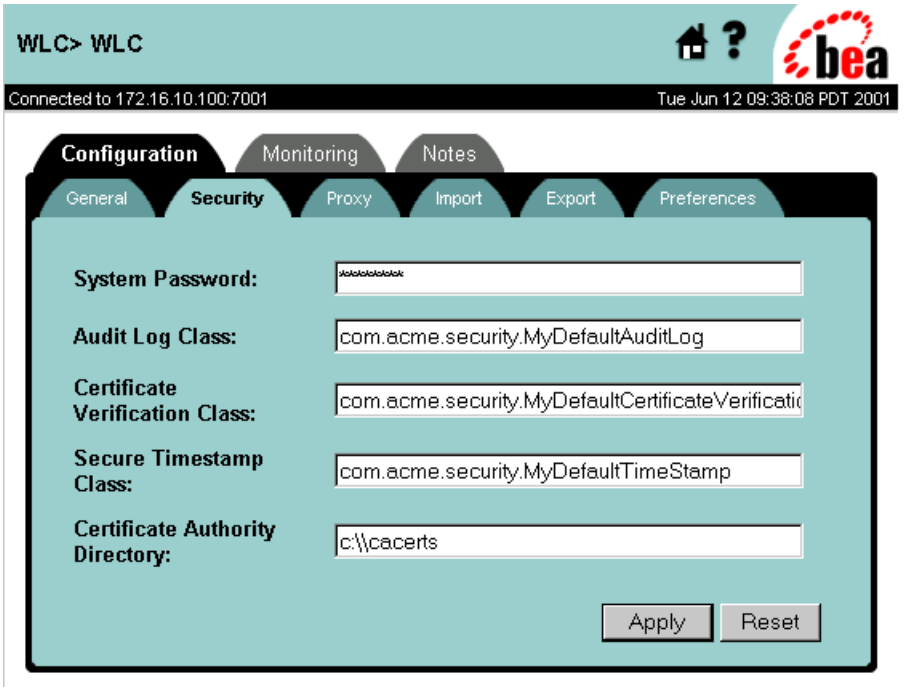
The WLC configuration tabs are displayed, as shown in the following figure.

Figure 3-9 WLC Server Configuration Tabs



3. Select the Security tab. The Security configuration page for the WebLogic Collaborate system is displayed, as shown in the following figure.

Figure 3-10 WebLogic Collaborate System Security Configuration Page



4. The following table describes the fields in the Security tab of the Configuration panel that you may need to configure. Note that the new configuration takes effect after the WebLogic Collaborate system is restarted.

Table 3-2 Configuring the WebLogic Collaborate System

Field	Description
System Password	Password for the WebLogic Collaborate system user. This is set when you install the WebLogic Collaborate software, and by default this password is `wlcsystem`. However, if you want to change it, you can enter a new password in this field.

Table 3-2 Configuring the WebLogic Collaborate System (Continued)

Field	Description
Audit Log Class	Java class that implements audit logging, which is used for nonrepudiation. You can use the audit log to reconstruct the sequence of events that have occurred during a conversation, along with the data exchanged. Depending on how you configure the audit log, the audit log may store each business message exchanged among trading partners along with digital signatures, timestamps, and other data. For more information about auditing, see "Secure Audit Log Service" on page 4-5.
Certificate Verification Class	Java class that calls out to software that verifies that a digital certificate submitted by a remote trading partner is valid. This class can call out to either the Online Certificate Status Protocol (OCSP) application that WebLogic Collaborate provides, or certificate verification provider software that you obtain from a trusted security vendor. For more information about the certificate verification class, see "Trading Partner Certificate Verification" on page 2-2.
Secure Timestamp Class	Java class that provides secure timestamping of business messages exchanged among trading partners. Timestamping is used for nonrepudiation. For more information about secure timestamping, see "Secure Timestamp Service" on page 4-3.
Certificate Authority Directory	Location that contains the Certificate Authorities of all the trading partner certificates configured in the WebLogic Collaborate repository.

Configuring Trading Partner Security

Configuring trading partner security involves setting the following for each trading partner:

- Certificates
- Transport security properties
- Document exchange security

- Delivery channel security

The following subsections describe how to configure trading partner security for each of these components.

Note: If you use the Bulk Loader to import data into the WebLogic Collaborate repository, the WebLogic Server users that represent each trading partner configured in the repository are not automatically created. You need to create these WebLogic Server users manually. For more information, see "Working with the Bulk Loader" in *Administering BEA WebLogic Collaborate*.

Configuring Trading Partner Certificates

WebLogic Collaborate provides a means to configure the following trading partner certificates.

Table 3-3 Trading Partner Certificates Configured in WebLogic Collaborate

Certificate	Description
Client certificate	Digital certificate of the remote or local trading partner. Configuring the client certificate is required when using the SSL protocol. **Certificate Details:** ■ Is of type X509 V3. ■ Is Privacy Enhanced Mail (PEM) or Definite Encoding Rules (DER) encoded. (The filename extension specifies the encoding type: `.pem` or `.der`.) ■ Is required for all trading partner types while using HTTPS. **Private Key Details:** ■ Is PEM or DER encoded. (The filename extension specifies the encoding type: `.pem` or `.der`.) ■ Is required only for local trading partner type. ■ Password protected private keys are not supported.

Table 3-3 Trading Partner Certificates Configured in WebLogic Collaborate

Certificate	Description
Server certificate	Digital certificate of the remote trading partner. Configuring the server certificate is required when using the SSL protocol. **Certificate Details:** ■ Is of type X509 V3. ■ Is PEM or DER encoded. (The filename extension specifies the encoding type: `.pem` or `.der`.) ■ Is required for remote trading partner types while using HTTPS.
Signature certificate	Certificate required of each trading partner if digital signature support, a requirement for nonrepudiation, is configured for the e-market. For a description of digital signature support, see "Digital Signature Support" on page 4-2. **Certificate Details:** ■ Is of type X509 V3. ■ Any encoding is allowed. ■ You use the RSA CertJ package to read the certificate. ■ Is required for all trading partner types using digital signature service. **Private Key Details:** ■ Uses only the PKCS8 format. ■ Is always password protected. (You specify the password as a system property in the `startweblogic` command.)

Table 3-3 Trading Partner Certificates Configured in WebLogic Collaborate

Certificate	Description
Encryption certificate	Certificate required of each trading partner when business message encryption is configured for the e-market. Note that encryption support is available only with the RosettaNet protocols. For a description of message encryption, see "Configuring Message Encryption" on page 3-31. **Certificate Details:** ■ Is of type X509 V3. ■ Any encoding is allowed. ■ Use RSA CertJ package to read the certificate. ■ Is required for all trading partner types using encryption service. **Private Key Details:** ■ Uses only the PKCS8 format. ■ Is always password protected. (You specify the password as a system property in the `startweblogic` command.)

Note the following general rules about configuring trading partner certificates:

- Each trading partner may have one client certificate and one server certificate, and any number of encryption and signature certificates.
- For each certificate, there is a trading partner type: Local or Remote. The contents of each certificate configuration tab depends on the trading partner type. For example, the tab for configuring a remote trading partner does not contain fields for entering information about private keys because information about private keys should be set only for local trading partners.
- For local trading partners, you do not configure a server certificate.
- When configuring a local trading partner, you do not need to provide a WebLogic Server username for that trading partner. The one exception to this rule is if the local trading partner is a trading partner lightweight client.
- Private keys for signature and message encryption certificates require a password. To set a password for a private key, specify the password as a system property on the command line that starts WebLogic Server. The following

example shows the command that starts WebLogic Server for the Hello Partner sample application.

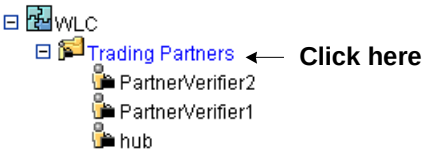
```
%JAVA_HOME%\bin\java -classic -ms64m -ms64m -classpath %START_WL_CLASSPATH%
 -Dbea.home=%BEA_HOME% -Dweblogic.home=%WL_HOME%
 -Dweblogic.system.home=%WLC_SAMPLES_HOME% -Dweblogic.Domain=samples
 -Dweblogic.management.password=security
 -Dcloudscape.system.home=%WLC_SAMPLES_CLOUDSCAPE_HOME% -Dweblogic.Name=myserver
 -Djava.security.policy=%WL_HOME%\lib\weblogic.policy
 -DKey.certificate-name.password=mypassword weblogic.Server
```

In the preceding example, **`certificate-name`** represents the name of the certificate for which a private key password is being specified, and **`mypassword`** represents the password.

To configure trading partner certificates, complete the following steps:

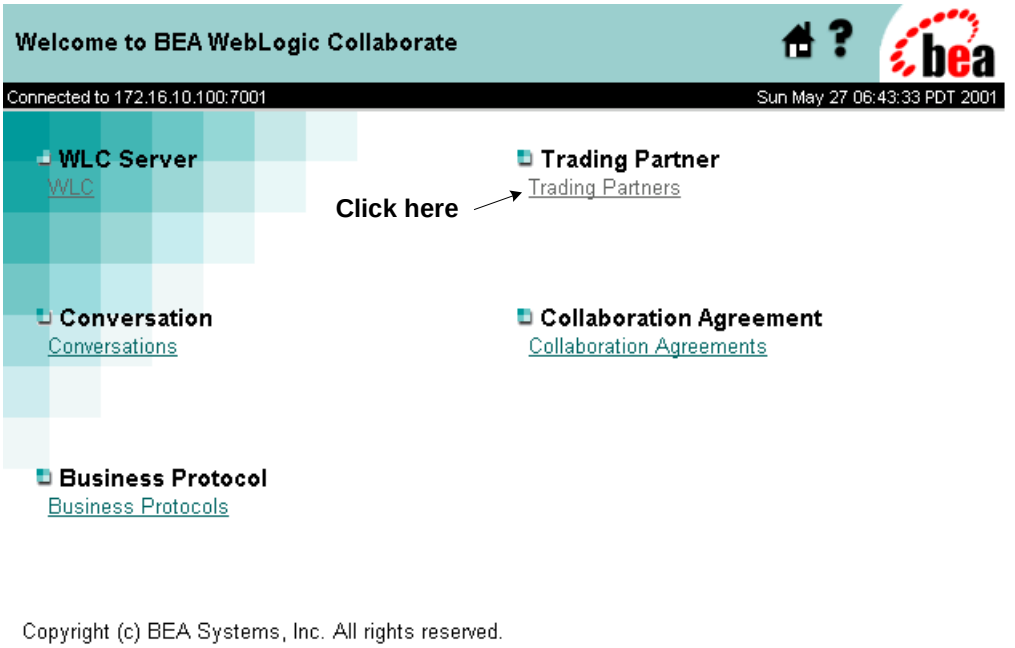
1. Display the main trading partner configuration page, which you can do one of the following ways:
   - Click Trading Partners in the navigation tree of the WebLogic Collaborate Administration Console.

Figure 3-11 Trading Partners Entry in the Navigation Tree



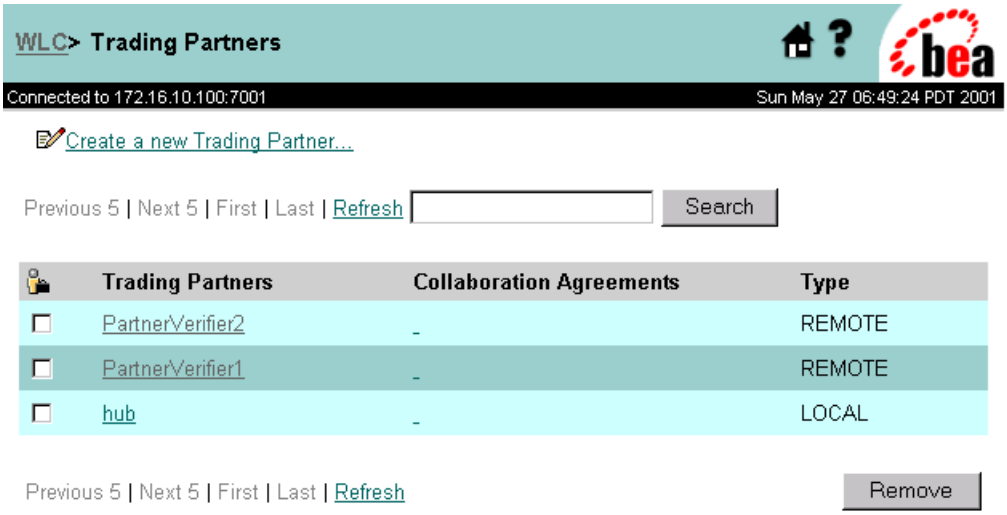
   - Click the Trading Partners link in the right pane, as shown in the following figure.

Figure 3-12 Accessing the Trading Partner Configuration Page



The main Trading Partners configuration page, where you can add, modify, and remove trading partners is shown in the following figure.

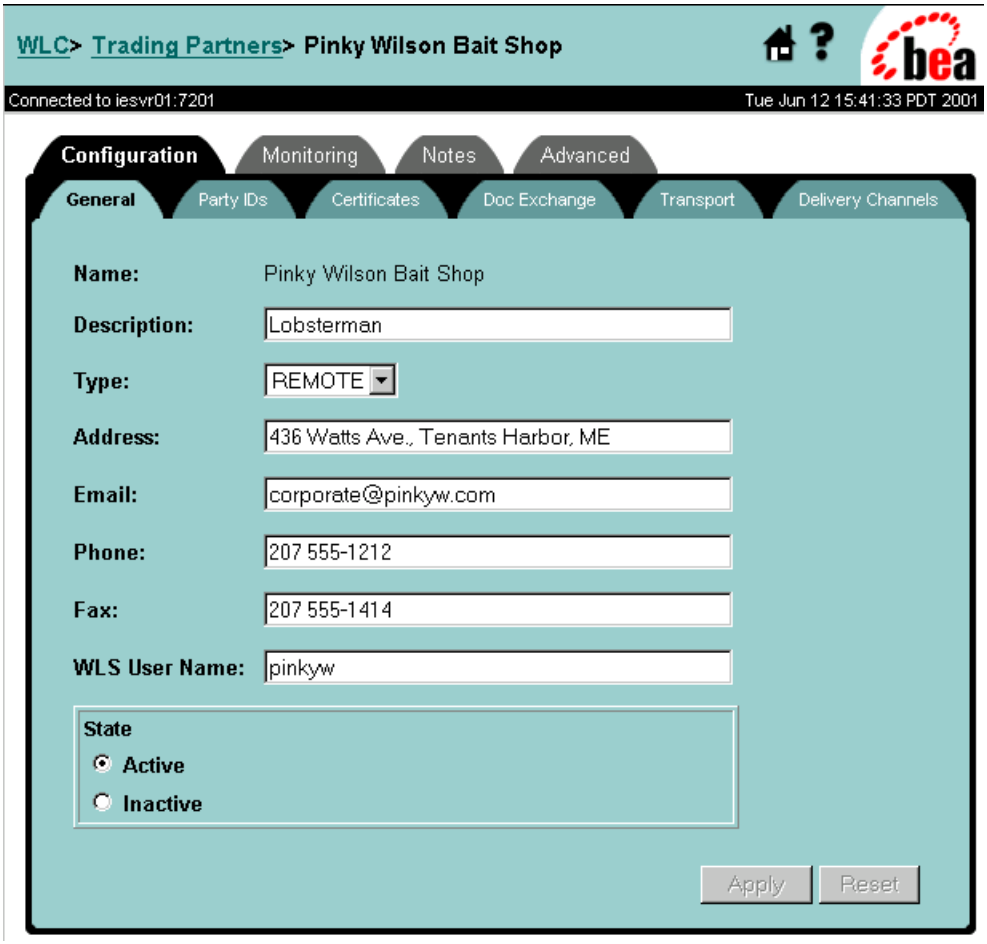
Figure 3-13 Main Trading Partner Configuration Page



Note: In the instructions that follow, we assume that the trading partner has already been created and configured, with the exception of security parameters. For complete details about configuring trading partners in general, see "Configuration Tasks" in *Administering BEA WebLogic Collaborate*.

2. Click the name of the trading partner whose security settings you want to configure. The General configuration page for the trading partner is shown in the following figure.

Figure 3-14 General Configuration Page for Trading Partner



3. Select the Certificates tab. The page on which you configure trading partner certificates is displayed, as shown in the following figure.

Figure 3-15 Trading Partner Certificates Configuration Page

Note: The preceding figure shows configuring a remote trading partner. If the trading partner were local, an additional field would be displayed showing the private key location for the certificate name.

4. To configure each of the trading partner certificates, complete the steps listed in the following table.

Table 3-4 Configuring Trading Partner Certificates

To configure . . .	Complete the following steps . . .
Client certificate	If you are configuring a local or remote trading partner: 1. In the Certificate Type selection box, select Client Certificate. 2. In the Certificate Name field, enter the client certificate name. 3. In the Certificate Location field, enter the filename and location on your WebLogic Collaborate machine where the client certificate is stored. 4. In the Private Key Location field, enter the filename and location on your WebLogic Collaborate machine where the private key of the local trading partner is stored. (This step applies only to local trading partners.) 5. Click Add/Apply.
Server certificate	If you are configuring a remote trading partner: 1. In the Certificate Type selection box, select Server Certificate. 2. In the Certificate Name field, enter the name of the server certificate for the remote trading partner's WebLogic Collaborate system. 3. In the Certificate Location field, enter the filename and location on your WebLogic Collaborate machine where the trading partner's server certificate is stored. 4. Click Add/Apply.
Signature certificate	For trading partners using digital signature support: 1. In the Certificate Type selection box, select Signature Certificate. 2. In the Certificate Name field, enter the signature certificate name. 3. In the Certificate Location field, enter the filename and location on your WebLogic Collaborate machine where the signature certificate is stored. 4. In the Private Key Location field, enter the filename and location on your WebLogic Collaborate machine where the local trading partner private key is stored. (This step applies only to local trading partners.) 5. Click Add/Apply.

Table 3-4 Configuring Trading Partner Certificates (Continued)

To configure . . .	Complete the following steps . . .
Encryption certificate	For trading partners using RosettaNet-based business message encryption: 1. In the Certificate Type selection box, select Encryption Certificate. 2. In the Certificate Name field, enter the encryption certificate name. 3. In the Certificate Location field, enter the location on your WebLogic Collaborate machine where the encryption certificate is stored. 4. In the Private Key Location field, enter the location on your WebLogic Collaborate machine where the local trading partner private key is stored. (This step applies only to local trading partners.) 5. Click Add/Apply.

Notes: When you create a trading partner in WebLogic Collaborate, a WebLogic Server user is created for that trading partner at run time using the WebLogic Server username that you specify. However, when you delete a trading partner from the WebLogic Collaborate repository, the corresponding WebLogic Server user is *not* automatically deleted. When you delete a trading partner, be sure also to manually delete the corresponding WebLogic Server user.

Visit the BEA Developer Center to obtain helpful resources, such as links to sites that provide useful tools for manipulating digital certificates and private keys, which you might find useful in managing WebLogic Collaborate security. You can reach the BEA Developer Center at the following URL:

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

Configuring a Secure Transport

When you configure a transport for a trading partner, you bind the trading partner's transport to a transport security protocol. For example, if a trading partner is configured to use SSL certificates, you must bind that trading partner's transport to a transport protocol that uses SSL. When a secure transport is configured, the client

certificate is used for outbound SSL. Because WebLogic Collaborate allows only one client certificate, there is no need to select the client certificate while configuring a secure transport.

To configure a secure transport for a trading partner, complete the following steps:

1. Select the Transport tab. The Transport configuration page is displayed. The top of this page is shown in the following figure.

Figure 3-16 Trading Partner Transport Configuration Page

2. Enter the information described in the following table.

Table 3-5 Configuring the Trading Partner Transport

Field	Description
Transport Name	The name of the trading partner transport. You can enter a name, or choose from the list of available transports displayed in the box labeled Available Transports. Note that each of the available transports has a security protocol bound to it, so if you choose from this list, the transport and security protocols are set automatically. For more information about specifying the transport name, see the online help for the Transport tab by clicking the question mark in the upper right.
Transport Protocol	The security protocol for the transport. You can choose between HTTP-1.1 and HTTPS-1.1. The HTTPS-1.1 protocol uses SSL. Note that if you choose HTTPS-1.1, the security protocol is displayed in the nonmodifiable field labeled Security Protocol.
URI Endpoint	The URI for the transport on the trading partner's WebLogic Collaborate system. To specify the URI endpoint, you can enter a URI in this field, or choose from one of the available URIs displayed in the box below this field. When you enter the URI endpoint, click Set, to establish the URI, or Remove, to clear an existing entry in the URI Endpoint field. For more information about specifying the URI endpoint, see the online help for the Transport tab by clicking the question mark in the upper right.

3. Click Add/Apply.

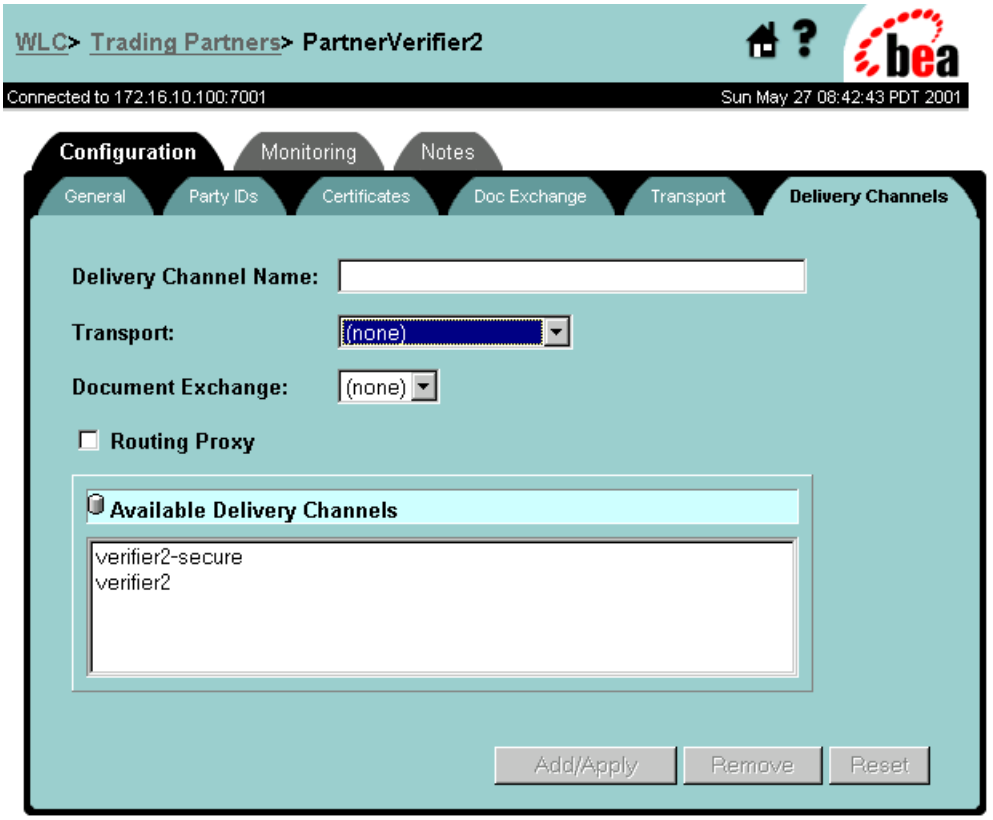
Configuring a Secure Delivery Channel

When you configure a trading partner's delivery channel, you have the option of making the delivery channel secure by binding it to the secure transport configured in "Configuring a Secure Transport" on page 3-23.

To configure a secure channel, complete the following steps:

1. Select the Delivery Channels tab. The Delivery Channels configuration page is displayed, as shown in the following figure.

Figure 3-17 Trading Partner Delivery Channels Configuration Page



2. Enter the information described in the following table.

Table 3-6 Configuring a Trading Partner Delivery Channel

Field	Description
Delivery Channel Name	The delivery channel name. You can enter a name in this field, or choose from the delivery channels listed in the Available Delivery Channels box below. For more information about specifying a delivery channel name, see the online help for the Delivery Channels page by clicking the question mark in the upper right.

Table 3-6 Configuring a Trading Partner Delivery Channel (Continued)

Field	Description
Transport	The name of the transport configured in the trading partner transport tab. This field gives you an opportunity to bind the delivery channel to a transport that you secured when configuring the transport properties, as described in "Configuring a Secure Transport" on page 3-23.
Document Exchange	The name of the document exchange to which you want to bind the delivery channel. For more information about binding a document exchange to a delivery channel, see the online help for the Delivery Channels page by clicking the question mark in the upper right.
Routing Proxy	Check this box if you want the trading partner delivery to act as a routing proxy (hub). For more information about proxy servers, see "Configuring WebLogic Collaborate to Use an Outbound HTTP Proxy Server" on page 3-39.

3. Click Add/Apply.

Configuring a Secure Document Exchange

When you configure the trading partner document exchange, you can associate a document exchange with a business protocol binding that provides digital signature support or message encryption. Digital signature support is available with all the business protocols supported in WebLogic Collaborate; however, message encryption is available only with the RosettaNet protocol.

To enable digital signature or message encryption support, complete the following steps:

1. Select the Document Exchange tab. The Document Exchange configuration page is displayed, as shown in the following figure.

Figure 3-18 Trading Partner Document Exchange Configuration Page

2. Enter the information described in the following table.

Table 3-7 Configuring a Trading Partner Document Exchange

In the field labeled . . .	Choose the following information . . .
Business Protocol Binding	The business protocol and version that supports the digital signature or message encryption capabilities that you want. The protocol you choose becomes bound to the trading partner document exchange identified at the top of the page.
Business Protocol Definition	The business protocol associated with the business protocol binding chosen in the preceding selection box.

3. For information about specifying data in the fields labeled Document Exchange Name, End Point Type, Confirmed Delivery, Message History, and Retries, see the online help for the Document Exchange page by clicking the question mark in the upper right.
4. For information about configuring digital signature information, see "Configuring Message Encryption" on page 3-29.
5. For information about configuring message encryption information, see "Configuring Digital Signatures for Nonrepudiation" on page 3-33.

Configuring Message Encryption

As mentioned in Chapter 1, "Introducing WebLogic Collaborate Security," the WebLogic Collaborate message encryption service encrypts business messages for the business protocols that require it. Currently, message encryption is supported only for the RosettaNet 2.0 protocol.

How WebLogic Collaborate Message Encryption Works

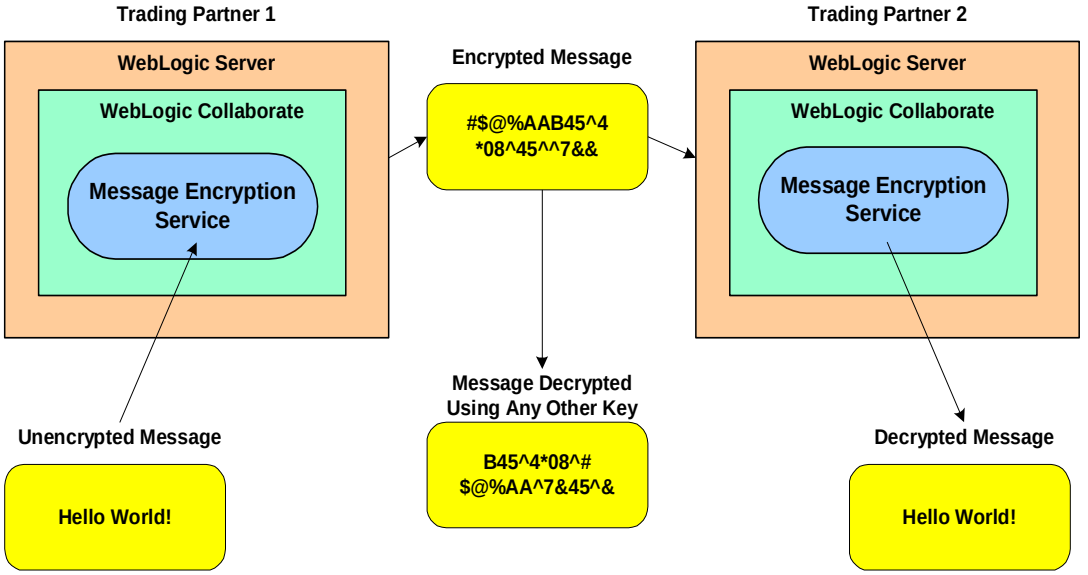
Data encryption works by using a combination of the sender's certificate, private key, and the recipient's certificate to encode a business message. The message can then be decrypted only by the recipient using the recipient's private key.

Note: The WebLogic Collaborate message encryption feature is controlled by licensing (Encryption/Domestic or Encryption/Export), but the decryption of a business message is not. If WebLogic Collaborate does not have a valid encryption license, WebLogic Collaborate disables the encryption service. However, WebLogic Collaborate can always decrypt business messages that are received.

The WebLogic Collaborate Release 2.0 message encryption service supports only the Rivest-Shamir-Adleman (RSA) encryption algorithm.

The following figure shows how data encryption is performed using the public and private keys.

Figure 3-19 WebLogic Collaborate Message Encryption Service



Note: To use message encryption, you must have a valid license for using the encryption service.

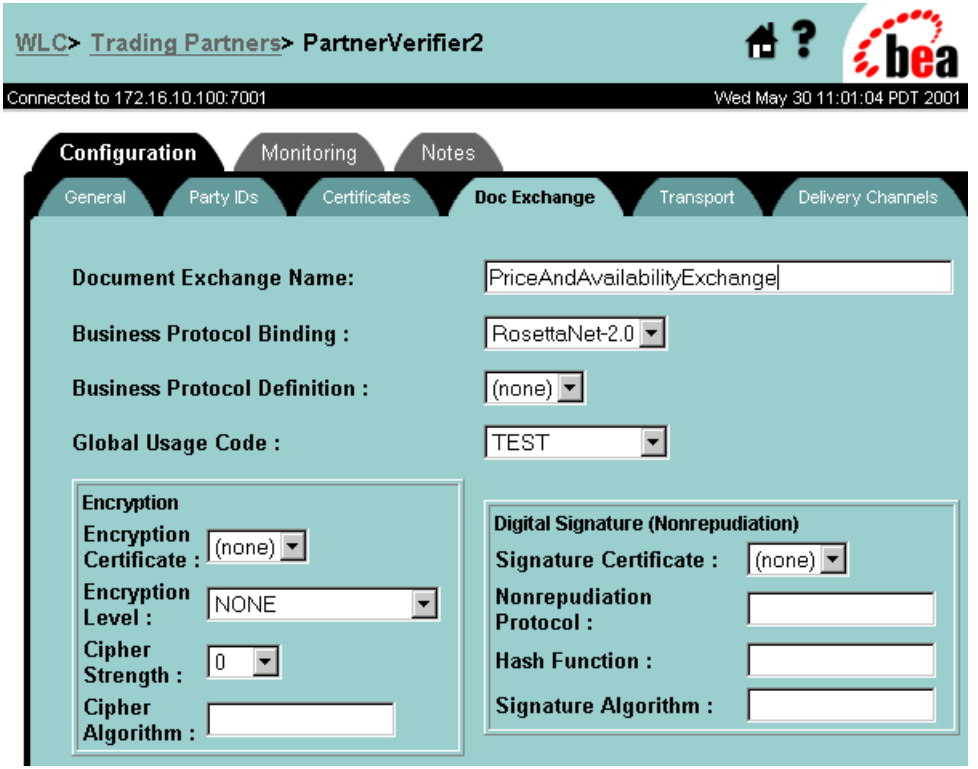
Configuring Message Encryption

To configure message encryption for business messages exchanged by trading partners in a RosettaNet 2.0-based conversation definition, complete the following steps:

1. Configure the trading partner as described in "Configuration Tasks" in *Administering BEA WebLogic Collaborate*.
2. Configure security for the trading partner delivery channel, as described in "Configuring a Secure Delivery Channel" on page 3-25. Be sure to configure the delivery channel using a transport that uses the appropriate RosettaNet 2.0 protocol binding.
3. Configure the trading partner document exchange, as described in "Configuring a Secure Document Exchange" on page 3-27. Be sure to configure the document exchange to support the appropriate RosettaNet 2.0 business protocol binding.

   Notice that when you select a RosettaNet business protocol binding on the Doc Exchange configuration page, the Encryption box is displayed in the lower left-hand corner of that configuration page. The following figure shows the Document Exchange configuration page with the Encryption box.

Figure 3-20 Configuration Box for Message Encryption on Doc Exchange Configuration Page



4. In the Encryption box, select the information described in the following table.

Table 3-8 Message Encryption Configuration Settings

In the field labeled . . .	Select the following . . .
Encryption Certificate	The name of the encryption certificate configured in "Configuring Trading Partner Certificates" on page 3-14.
Encryption Level	The parts of the business message that you want to have encrypted. Choose `PAYLOAD` if you want to encrypt only the XML business document(s) part of the message. Choose `ENTIRE_PAYLOAD` if you want to encrypt the business documents and all attachments in the message.

Table 3-8 Message Encryption Configuration Settings (Continued)

In the field labeled . . .	Select the following . . .
Cipher Strength	Either 56- or 128-bit encryption. Note that 128-bit encryption is not available in some localities.

Note that the field labeled Cipher Algorithm is a nonmodifiable information field containing the name of the algorithm. With Release 2.0 of WebLogic Collaborate, the only value displayed in this field is RSA.

5. Click Add/Apply.

Configuring Digital Signatures for Nonrepudiation

Digital signature support (described in detail in Chapter 4, "Implementing Nonrepudiation") provides a means to prevent anyone or anything from tampering with the contents of a business message, especially when the business message is in transit between two trading partners. Digital signature support is a requirement for nonrepudiation.

If you are implementing nonrepudiation, you need to configure digital signature support in the WebLogic Collaborate Administration Console, which you can do by completing the following steps:

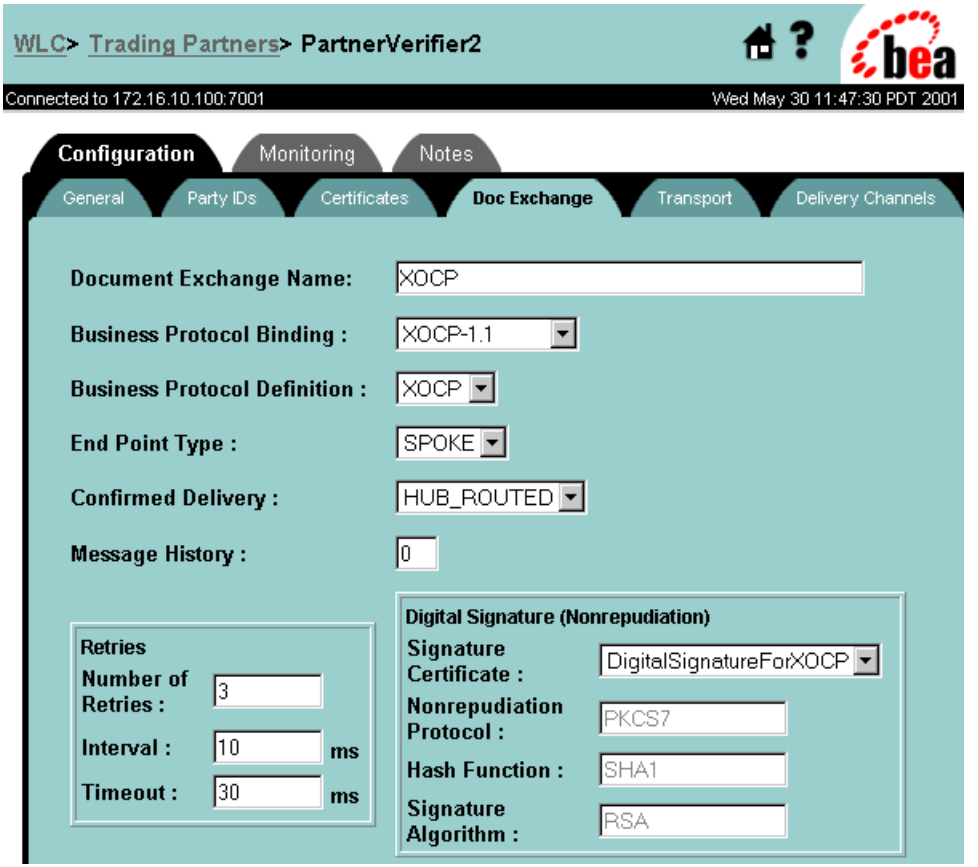
1. Configure the trading partner, as described in "Configuration Tasks" in *Administering BEA WebLogic Collaborate*.
2. Configure the trading partner signature certificate, as described in "Configuring Trading Partner Certificates" on page 3-14.
3. Configure the trading partner delivery channel security, as described in "Configuring a Secure Delivery Channel" on page 3-25. Be sure to configure the delivery channel using a transport that uses the appropriate protocol binding.

4. Configure the trading partner document exchange, as described in “Configuring a Secure Document Exchange” on page 3-27. Be sure to configure the document exchange to support the appropriate business protocol binding.

5. In the Doc Exchange tab, notice the box labeled Digital Signature (Nonrepudiation) in the lower right. In this box, choose the trading partner signature certificate identified in “Configuring Trading Partner Certificates” on page 3-14.

   When you choose a signature certificate, notice the data displayed in the nonmodifiable fields that are associated with the signature certificate, as shown in the lower right in the following figure.

Figure 3-21 Configuring Nonrepudiation



These nonmodifiable fields are used for the following purposes.

- Nonrepudiation protocol—identifies the business protocol associated with the signature certificate.
- Hash Function—identifies the function used for encrypting passwords exchanged among trading partners. The hash function used by both the RosettaNet and XOCP protocols in WebLogic Collaborate is `SHA1`.

- Signature Algorithm—identifies the algorithm used for encrypting the signature certificates exchanged among trading partners. The signature algorithm used by both the RosettaNet and XOCP protocols in WebLogic Collaborate is `RSA`.

Customizing the WLCCertAuthenticator Class

The `WLCCertAuthenticator` class is an implementation of the WebLogic Server `CertAuthenticator` class. The default implementation of the `WLCCertAuthenticator` class maps the digital certificate of the trading partner to the corresponding trading partner user defined in the WebLogic Collaborate repository. 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 WebLogic Server user.

The `WLCCertAuthenticator` class is invoked by WebLogic Server after an SSL connection between the trading partner and WebLogic Server has been established. The class can extract data from a digital certificate to determine the trading partner name that corresponds to the digital certificate.

The following code example, in which the WebLogic default realm for retrieving users is used, shows how the `WLCCertAuthenticator` class is customized:

```
public User authenticate(String userName, Certificate[] certs, boolean ssl)
{

String user = null;

// If not using SSL, return
if (ssl == false)
{
return null;
}

// Verify that the certificate is either a c-hub certificate or a trading partner
// certificate, then return the corresponding WLS user.

if ((user = Security.isValidWLCCertificate(certs))!= null)
{
return realm.getUser(user);
}
```

```
// Certificate is not a valid WLC certificate.
// Check here for non-WLC certificate and return the corresponding user.
}
```

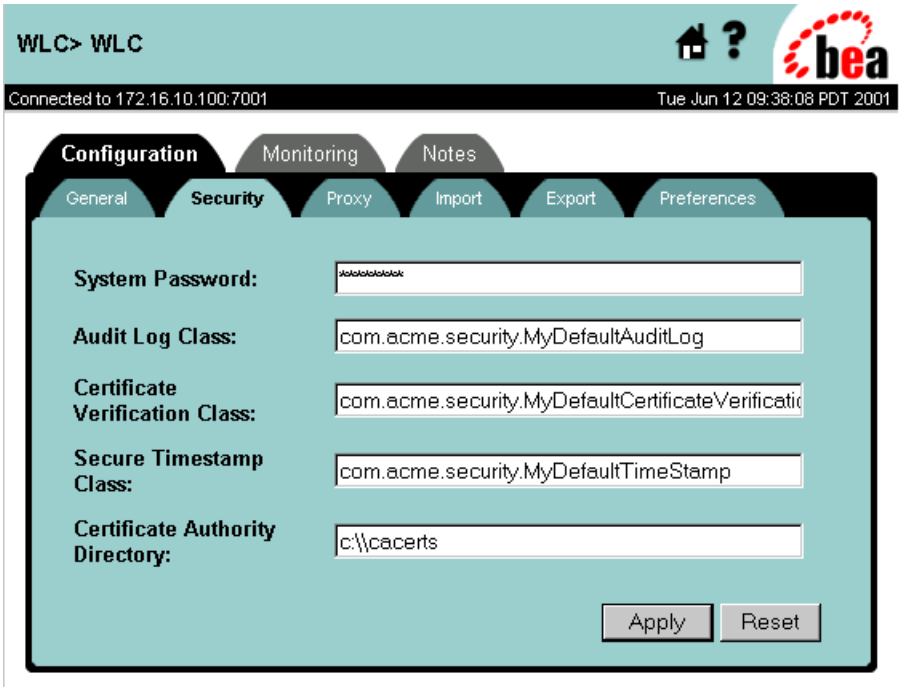
Configuring a Certificate Verification Provider Interface

As explained in "Trading Partner Certificate Verification" on page 2-2, you use a certificate verification provider to validate a trading partner's digital certificate. If you are using a certificate verification provider (CVP), you need to configure it in the WebLogic Collaborate Administration Console, using the steps described in this section.

To configure a CVP:

1. Start the WebLogic Collaborate Administration Console.
2. In the main page of the WebLogic Collaborate Administration Console, click the link under WLC Server, as described in "Configuring Security for the WebLogic Collaborate System" on page 3-9.
3. In the WLC Server Configuration panel, select the Security tab. This displays the page shown in the following figure.

Figure 3-22 WebLogic Collaborate System Security Configuration Page



4. In the field labeled Certificate Verification Class, enter the fully qualified name of the Java class that implements the CVP.
5. Click Apply.

Note: You can load a certificate verification provider via the Bulk Loader. For more information, see "Working with the Bulk Loader" in *Administering BEA WebLogic Collaborate*.

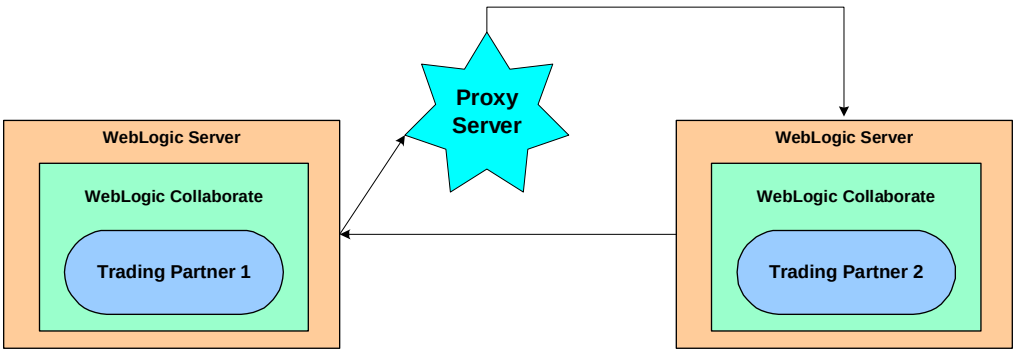
Configuring WebLogic Collaborate to Use an Outbound HTTP Proxy Server

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

- Hide, from external hackers, the local network addresses of the WebLogic Servers that host WebLogic Collaborate
- Restrict access to the external network
- Monitor external network access to the WebLogic Servers that host WebLogic Collaborate

When proxy servers are configured on the local network, 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.

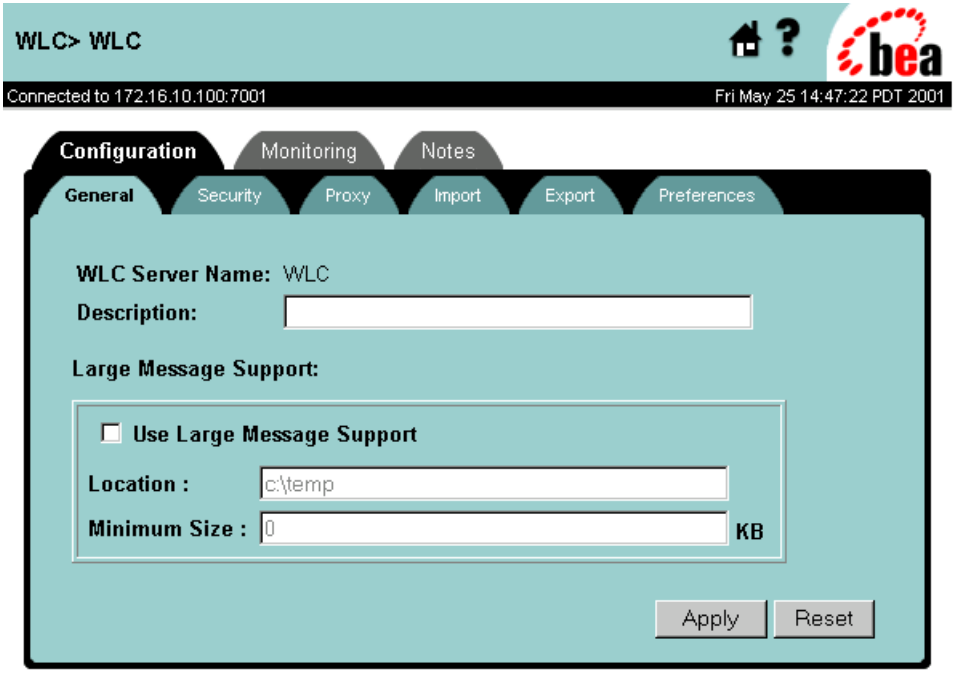
Figure 3-23 Proxy Server



To configure a proxy server for WebLogic Collaborate, complete the following steps:

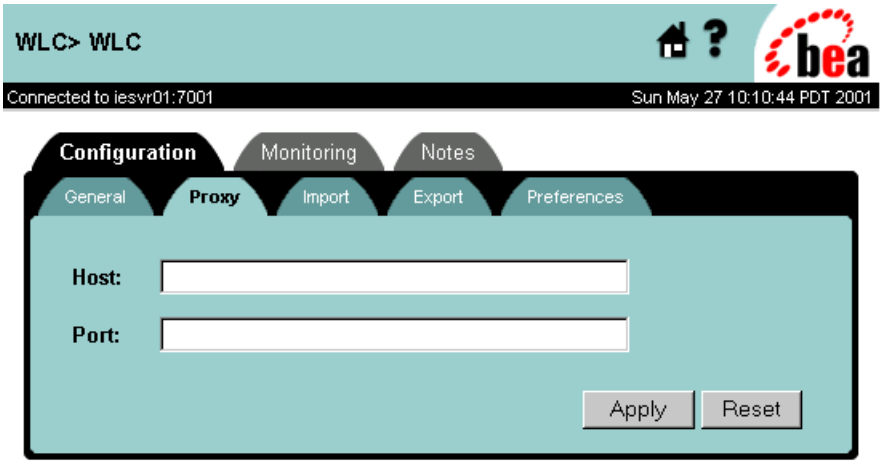
1. Display the configuration tabs in the right pane of the WebLogic Collaborate Administration Console window, as shown in the following figure.

Figure 3-24 Configuration Tabs in the WebLogic Collaborate Administration Console



2. Select the Proxy tab. The Proxy configuration page is displayed, as shown in the following figure.

Figure 3-25 WebLogic Collaborate Proxy Server Configuration Page



3. In the field labeled Host, enter the address of the proxy server used for the WebLogic Collaborate server, if any. For example:

   `myproxy.mycompany.com.`

4. In the field labeled Port, enter the port number for the proxy server.

5. Click Apply.

6. 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";
```

Configuring WebLogic Collaborate with a Webserver and a WebLogic Proxy Plug-In

You can configure WebLogic Collaborate with a webserver, such as Apache server, that is programmed to service business messages from a remote trading partner. The webserver can provide the following services:

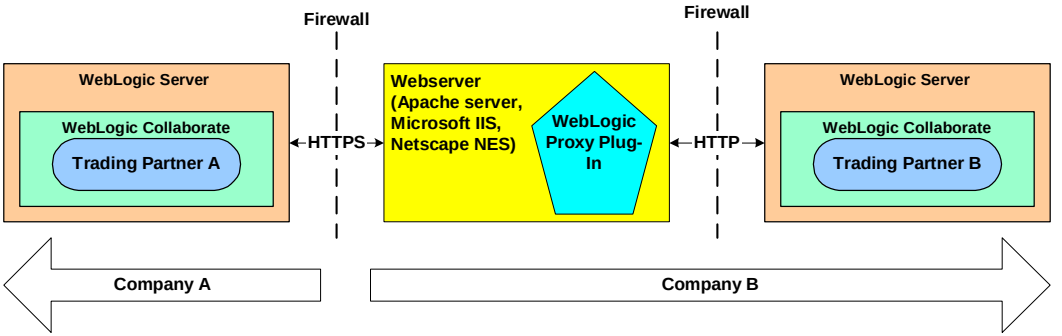
- Receive the business messages from the remote trading partner
- Authenticates the trading partner digital certificate

The webserver uses the WebLogic proxy plug-in, which you can configure to provide the following services:

- Forwards business messages received by the webserver to WebLogic Collaborate, which is running inside a secure internal network.
- Extract the remote trading partner certificate from the webserver and forward it to WebLogic Server for authentication. WebLogic Collaborate can then authenticate the trading partner certificate and business message.

The following figure shows the topology of an environment that uses a webserver, the WebLogic proxy plug-in, and WebLogic Collaborate.

Figure 3-26 Using a Webserver and the WebLogic Proxy Plug-In



Configuring the Webserver

To configure the webserver, see "Deploying and Configuring Web Applications" in the *BEA WebLogic Server Administration Guide*.

The following code example provides the segment of `httpd.conf` (for Apache server) for configuring the proxy plug-in:

```
# LoadModule foo_module libexec/mod_foo.so
LoadModule weblogic_module    libexec/mod_wl_ssl.<suffix>

<Location /weblogic>
 SetHandler weblogic-handler
 PathTrim /weblogic
 WebLogicHost myhost
 WebLogicPort 80
</Location>
```

Note that in WebLogic Server 6.0, the proxy plug-in supports only one-way SSL. Because WebLogic Server hosting WebLogic Collaborate is configured with mutual authentication, it is important that you do *not* configure the proxy plug-in with SSL.

WebLogic Server User Identity for the Trading Partner

The WebLogic Server user identity is optional when you configure the remote trading partner. If a particular WebLogic Collaborate deployment has stringent security requirements, we recommend the following:

- Configure the ACLs for the transport servlet to enable permissions for the WebLogic Server users that map to the remote trading partner certificates.
- Disable guest users so that users with unknown or invalid certificates are unable to enter the WebLogic Server system.

Configuring WebLogic Process Integrator Access to the WebLogic Collaborate Repository

If you use WebLogic Collaborate with WebLogic Process Integrator, note the following configuration tasks for sharing access to the WebLogic Collaborate repository.

- You need to configure WebLogic Process Integrator to have permissions to use the WebLogic Collaborate repository. You can do this by adding the WebLogic Server group `wlpiUsers` to the ACL for the JDBC connection pool used by the WebLogic Collaborate repository.
- If a user of the WebLogic Process Integrator components Studio or Worklist needs access to workflow templates stored in the WebLogic Collaborate repository, you need to add that user to the appropriate ACLs for the WebLogic Server administration MBeans.

  You can do this by specifying the following ACLs on the WebLogic Server MBeans for the user, where *user* represents the name of the WebLogic Process Integrator user:

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

For information about configuring ACLs for WebLogic Collaborate resources, see “Configuring Access Control Lists for WebLogic Collaborate” on page 3-6.

4 Implementing Nonrepudiation

This topic includes the following sections:

- Overview of Nonrepudiation
- Using the Service Provider Interfaces (SPIs) for Nonrepudiation

Overview of Nonrepudiation

Nonrepudiation is the ability of a trading partner to prove or disprove having previously sent or received a particular business message to or from another trading partner. Consider the following example.

Trading Partner A has agreed to purchase 1000 ergonomic chairs from Trading Partner B. In the course of this agreement, Trading Partner A has sent a business message to Trading Partner B agreeing to buy the chairs at a set price. Later, though, Trading Partner A disputes the original price and denies having sent a message in which they agreed to pay that price.

If a reliable nonrepudiation system has been in place, Trading Partner B can disprove Trading Partner A's claim by producing a document from Trading Partner A specifying the amount Trading Partner A agreed to pay. Further, if this original document is digitally signed, timestamped, recorded, and secured by a trusted third-party source, the validity of this document has full legal recourse.

Nonrepudiation, or the ability to provide legal evidence of the involvement of a denying party, is a requirement for critical business messages. WebLogic Collaborate supports both nonrepudiation of origin and nonrepudiation of receipt:

- Nonrepudiation of origin links the business message and the sender of the message. It provides legal evidence that you have sent a business message.
- Nonrepudiation of receipt links the business message and the recipient of the message. It provides legal evidence that you have received a business message.

To support nonrepudiation, the WebLogic Collaborate software incorporates the following services:

- Digital signatures
- Secure timestamp
- Secure audit log

The remaining sections in this topic describe each of these services and explain how to incorporate them into your WebLogic Collaborate environment.

Digital Signature Support

The purpose of digital signature support is to provide a means to prevent anyone or anything from tampering with the contents of a business message, especially when the business message is in transit between two trading partners. WebLogic Collaborate provides digital signature support that conforms to the Public Key Cryptography Standard 7 (PKCS7) packaging for digital signatures.

A digital signature itself is a set of data appended to a business message consisting of an encrypted, one-way hash value of data packaged in a specific format (for example, PKCS7 SignedData). A digital signature:

- Validates that the contents of a digitally signed message have not been tampered with.
- Contains the identity of the sender of the business message.

The data required to create a digital signature is obtained from the trading partner configuration data in the repository. The information required to create a digital signature also includes the following:

- Trading partner signature certificate and private key
- Certificate authority certificate for the trading partner signature certificate
- Hash algorithm name: SHA1
- Signature algorithm name: RSA

Business Protocols with Which You May Use Digital Signature Support

WebLogic Collaborate provides digital signature support for messages that use the following business protocols:

- RosettaNet 1.1
- RosettaNet 2.0
- XOCP 1.1

Configuring Digital Signature Support

When you configure the WebLogic Collaborate Server, you have the option of specifying a digital signature service. To use a digital signature service, you must configure it as described in "Configuring Digital Signatures for Nonrepudiation" on page 3-33.

Secure Timestamp Service

If nonrepudiation is being used, secure timestamp services are required to attach a Coordinated Universal Time (UTC) timestamp to the secure audit log when business messages are also logged to the secure audit log. For example, when you receive a business message, a timestamp is entered as a nonrepudiation of receipt (NRR) message in the audit log. When you send a business message, a timestamp is entered as a nonrepudiation of origin (NRO) message in the audit log. WebLogic Collaborate includes a Service Provider Interface (SPI) so that you can incorporate a secure timestamp service from a trusted third-party provider.

If you incorporate a secure timestamp service from a trusted third-party provider, you need to create a Java class file that implements the `com.bea.b2b.security.TimestampProvider` interface. In the class methods (for

example, `getTimestamp`) of your class implementing the `com.bea.b2b.security.TimestampProvider` interface, you call out to the third party timestamp provider. For details about creating this application, see "Using the SPI for the Secure Timestamp Service" on page 4-11.

WebLogic Collaborate prohibits more than one secure timestamp provider from being registered in WebLogic Collaborate. This restriction ensures that all timestamps created in the WebLogic Collaborate system are ordered chronologically.

Note: If you do not configure a secure timestamp service provider in your WebLogic Collaborate server, system time is used for timestamping system events and signatures.

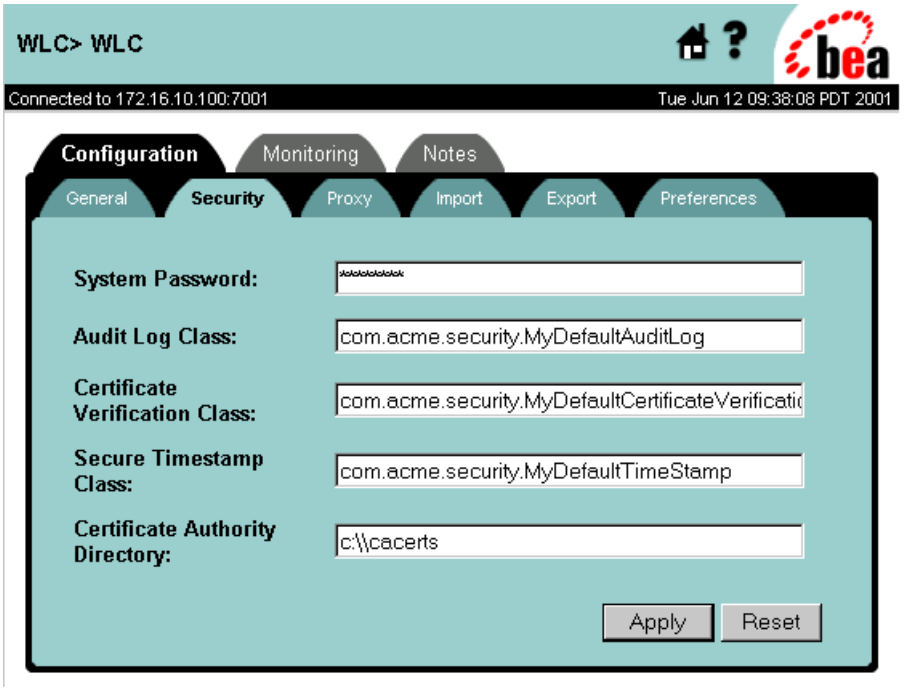
For details about the secure timestamp SPI, see "Using the SPI for the Secure Timestamp Service" on page 4-11.

Configuring the Secure Timestamp Service

To configure the secure timestamp service, complete the following steps:

1. Start the WebLogic Collaborate Administration Console and display the WLC Server configuration page, as described in "Configuring Security for the WebLogic Collaborate System" on page 3-9.
2. Select the Security tab. The WebLogic Collaborate Security configuration page is displayed, as shown in the following figure.

Figure 4-1 WebLogic Collaborate System Security Configuration Page



3. In the field labeled Secure Timestamp Class, enter the fully qualified name of the Java class that implements the secure timestamp interface.
4. Restart WebLogic Server so that the new configuration takes effect.

Secure Audit Log Service

A secure audit log is also required for nonrepudiation. This log typically stores each business message with its digital signature and secure timestamp. You use an audit log to reconstruct the sequence of messages and other system events that have occurred during the exchange of business messages among trading partners.

As with the timestamp service, WebLogic Collaborate provides a Service Provider Interface (SPI) for you to configure a trusted, third-party provider of the secure audit log. If you incorporate a secure audit log service from a trusted third-party provider,

you need to create a class file that implements the `com.bea.b2b.security.AuditLogProvider` interface. In the class methods of your class implementing the `com.bea.b2b.security.AuditLogProvider` interface (for example, `log`), you call out to the third party audit log provider. For details about creating this implementation, see "Using the SPI for the Secure Audit Log" on page 4-12.

Note: If you do not configure a third-party provider for a secure audit log service, WebLogic Collaborate provides a default audit log in a file named `secureaudit.log`, which you can enable by setting the system property `bea.secureaudit` to `on`. This file is based on the logging subsystem in WebLogic Collaborate, and is protected by only the underlying operating system's file permissions system. This file is not digitally signed or encrypted.

Writing to the Audit Log Directly

As an alternative to writing a Java implementation of the `com.bea.b2b.security.AuditLogProvider` interface to call out to an application that writes to the audit log, you can write an application that writes to the audit log directly via an invocation to the `com.bea.b2b.security.Audit.log(byte[] data)` method, as shown in the code example provided in this section.

This example is a modification of the `CreateMultiplyReply.java` class, which is located in the following directory, where `WLC_HOME` represents the directory in which the WebLogic Collaborate software is installed:

- **Windows**

  `%WLC_HOME%\config\samples\messageManipulators`

- **UNIX**

  `$WLC_HOME/config/samples/messageManipulators`

In this example, the bolded code shows the statements that have been added to show writing to the audit log.

Listing 4-1 Example of Writing to the Audit Log Directly

```
package wlcsamples;

import java.io.*;

import org.apache.xerces.dom.*;
import org.w3c.dom.*;

import com.bea.eci.logging.*;
import com.bea.b2b.wlpi.MessageManipulator;
import com.bea.b2b.wlpi.WorkflowInstance;
import com.bea.b2b.wlpi.WLPIException;

import com.bea.b2b.protocol.conversation.ConversationType;
import com.bea.b2b.enabler.*;
import com.bea.b2b.enabler.xocp.*;
import com.bea.b2b.protocol.messaging.*;
import com.bea.b2b.protocol.xocp.conversation.local.*;
import com.bea.b2b.protocol.xocp.messaging.*;

//Import the Audit class from security package.
import com.bea.b2b.security.Audit;


public class CreateMultiplyReply implements MessageManipulator{

  public CreateMultiplyReply(){};

  public XOCPMessage manipulate(WorkflowInstance instance,
                                XOCPMessage in)
    throws WLPIException{
    debug("In CreateMultiplyReply");

    int integerOne =
      ((Long)instance.getVariable(WLCSamplesConstants.INTEGER_ONE_VAR)).intValue
();
    int integerTwo =
      ((Long)instance.getVariable(WLCSamplesConstants.INTEGER_TWO_VAR)).intValue
();
    int result = integerTwo * integerOne;

    debug("integerOne = " + integerOne);
    debug("integerTwo = " + integerTwo);
    debug("result = " + result);

    String sender =
      ((String)instance.getVariable(WLCSamplesConstants.SENDER_VAR));
```

```
String recip =
  ((String)instance.getVariable(WLCSamplesConstants.RECIPIENT_VAR));

debug("sender = " + sender);
debug("recip = " + recip);

XOCPMessage xocpmsg = null;
try{
  DOMImplementationImpl domi = new DOMImplementationImpl();

  // "reply" - (param1) The qualified name of the document
  //           type to be created.
  // "reply" - The document type public identifier.
  // "multiply-reply.dtd" - The document type system identifier
  DocumentType dType = domi.createDocumentType("reply", "reply",
                                          "multiply-reply.dtd");

  org.w3c.dom.Document rq = new DocumentImpl(dType);
  Element root = rq.createElement("multiply-reply");
  rq.appendChild(root);

  Element elementProduct = rq.createElement("integer-product");
  Text tProduct = rq.createTextNode( new Integer(result).toString() );
  elementProduct.appendChild(tProduct);
  root.appendChild(elementProduct);

  String note =
    "Dear " + sender + ": " +
    "Here is the product of " + integerOne + " and " + integerTwo + ".  " +
    "With Love, " + recip + ".";

  debug("NOTE...\n" + note );

  // we got the data here. Let us log it
  new byte[] ba = note.getBytes();
  Audit.log(ba);

  Element elementNote = rq.createElement("note");
  Text tNote = rq.createTextNode( note );
  elementNote.appendChild(tNote);
  root.appendChild(elementNote);


  debug("Created root: \n" + root.toString() );

  xocpmsg = new XOCPMessage("");
  xocpmsg.addPayloadPart(new BusinessDocument(rq));

}catch(Exception e){
```

```
      debug("Error at manipulate.");
      e.printStackTrace();
      throw new WLPIException("CreateMultiplyReply raised exception:" + e);
    }

    return xocpmsg;
  }

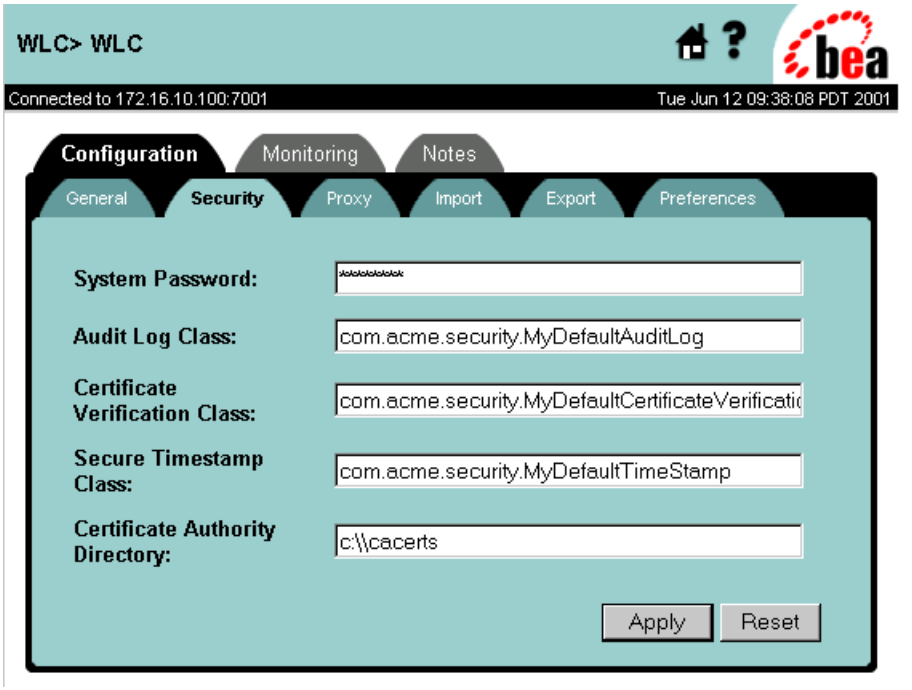
  /**
   * A simple routine that writes to the wlc log
   */
  private static String debug(String msg){
      UserLog.log("****CreateMultiplyReply: "+msg);
      return msg;
  }
}
```

Configuring the Secure Audit Log

To configure the secure audit log, complete the following steps:

1. Start the WebLogic Collaborate Administration Console and display the WLC Server configuration page, as described in “Configuring Security for the WebLogic Collaborate System” on page 3-9.
2. Select the Security tab. The WebLogic Collaborate Security configuration page is displayed, as shown in the following figure.

Figure 4-2 WebLogic Collaborate System Security Configuration Page



3. In the field labeled Audit Log Class, enter the fully qualified name of the Java class that implements the secure audit log.
4. Restart WebLogic Server so that the new configuration takes effect.

Using the Service Provider Interfaces (SPIs) for Nonrepudiation

This section describes the SPIs for the following nonrepudiation services:

- Secure Timestamp Service
- Secure Audit Log Service

Using the SPI for the Secure Timestamp Service

WebLogic Collaborate allows you to create a customized secure timestamp service by implementing the `com.bea.security.TimeStampProvider` interface. If you implement a timestamp using the SPI described in this section, you must configure this service later in the WebLogic Collaborate Administration Console so that the service is invoked properly during run time.

The `com.bea.b2b.security.TimeStampProvider` interface has the following methods, which a timestamp application must implement:

- `String getTimestamp()`

  This method returns a string specifying the time in Coordinate Universal Time (UTC) format.

- `long getTimestampInMillis()`

  This method returns a string specifying the UTC time in milliseconds.

Your implementation of the timestamp interface must include a default public constructor with no arguments. Neither the constructor nor any methods in the class that implements the `TimeStampProvider` interface should throw any exceptions.

Using the SPI for the Secure Audit Log

WebLogic Collaborate allows you to create a secure audit log service by implementing the `com.bea.security.AuditLogProvider` interface. If you implement an audit log service using the SPI described in this section, you must configure this service later in the WebLogic Collaborate Administration Console so that the service is invoked properly during run time.

The `com.bea.b2b.security.AuditLogProvider` interface has the following methods, which a secure audit log application must implement:

- `void init()`

  This method initializes the audit log.

- ```
  void log (java.lang.String component,
            java.lang.String type,
            byte[] data,
            java.lang.String principal)
  ```

  This method is invoked to log a message in the secure audit log. It has the following parameters:

  - `java.lang.String component`

    Contains the component that is logging the message

  - `java.lang.String type`

    Specifies the type of the nonrepudiation message

  - `byte[] data`

    Contains the data to be logged

  - `java.lang.String principal`

    Contains the name of the trading partner who is logging this message

Your implementation of the secure audit interface must include a default public constructor with no arguments. Neither the constructor nor any methods in the class that implements the `AuditLogProvider` interface should throw any exceptions.
  ```

Audit Log Messages

All log messages correspond to the DTD `log-message.dtd`, which defines the contents for each message type.

All audit log messages have the following three identifiers:

- Location—the location, in WebLogic Collaborate, in which the message is stored
- Type—the message type
- Data—the actual information that is being logged

The following table describes the contents of the data for each of the message types. All the log messages contain the timestamp obtained from the timestamp provider that is configured in WebLogic Collaborate.

Message Type	Description
NRR	Nonrepudiation of receipt. Contains that name of the trading partner receiving the business message and the application data.
NRO	Nonrepudiation of origin. Contains the name of the trading partner sender, the business message, and the application data.
APP	Is logged from any trading partner Java class via the `Audit.log(byte[] data)` method.The data format for this message type is any stringified XML document. Because the application is logging the message, the contents of the data are controlled by the application itself.

Audit Log DTD

The following code example shows the `log-message.dtd` file:

```
<!ELEMENT LOG (non-repudiation-origin| non-repudiation-receipt | application)>
<!ATTLIST LOG  time-stamp CDATA  #REQUIRED >
<!ATTLIST LOG location CDATA  #IMPLIED >
<!ATTLIST LOG Principal CDATA  #IMPLIED >
<!ELEMENT non-repudiation-origin (#PCDATA)>
<!ELEMENT non-repudiation-receipt (#PCDATA)>
<!ELEMENT application (#PCDATA)>
```

A Using the Secure Fingerprint Utility

Certificates are used to authenticate trading partners in WebLogic Collaborate. When you configure the WebLogic Collaborate Server, you specify a Certificate Field Name and a Server Certificate Field Name. A Certificate Field name is used for mapping trading partner certificates to WebLogic Server users, and a Server Certificate Field name is used for authenticating a remote SSL server.

Choose one of the following Certificate Field Names: None, Email, or Fingerprint. When you choose Fingerprint, you extract the fingerprint value from the digital certificate and enter it in the Certificate Field Value.

A WebLogic Collaborate utility is provided in the `<WLC_HOME>/bin` directory to simplify the task of extracting the fingerprint value from the digital certificate.

The syntax for the fingerprint utility is platform-dependent.

Windows

```
prompt> cd %WLC_HOME%
prompt> setenv.cmd
prompt> fingerprint.cmd certificate_file
```

UNIX

```
prompt> cd $WLC_HOME
prompt> . ./setenv.sh
prompt> fingerprint.sh certificate_file
```

Here *`certificate_file`* is the name of the file containing the digital certificate.

The utility returns the MD5 fingerprint value in the form of a hexadecimal ASCII string from the digital certificate.

Index

D

E

F

T

U

W