BEA WebLogic Integration™® and Cyclone Interchange

Solution Guide

BEA Confidential. For Internal Company Use Only.

Version 8.14
Document Revised: August 1, 2005

Copyright

Copyright © 2005 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

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

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

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

Trademarks or Service Marks

BEA, BEA JRockit, BEA Liquid Data for WebLogic, BEA WebLogic Server, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA MessageQ, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic JRockit, BEA WebLogic Log Central, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server Process Edition, BEA WebLogic WorkGroup Edition, BEA WebLogic Workshop, and Liquid Computing are trademarks of BEA Systems, Inc. BEA Mission Critical Support is a service mark of BEA Systems, Inc. All other company and product names may be the subject of intellectual property rights reserved by third parties.

All other trademarks are the property of their respective companies.

BEA Confidential Information

BEA Confidential. Do not copy or distribute without express permission by BEA Systems, Inc.
This document is available on a controlled release basis or for use internal to BEA Systems, Inc.

Contents

About This Document

Integrating BEA WebLogic Integration and Cyclone Interchange

Index

About This Document

This document describes various options for trading ebXML and EDI documents among B2B collaborators using WebLogic Integration (versions 8.1.4, 8.1.5, and 8.5) and Cyclone Interchange (version 5.3.2) in an integrated environment. This document describes a range of common trading scenarios, from single end-point solutions to advanced clustered configurations that offer fault tolerance and high availability. This document focuses on ways to configure WebLogic Integration and Cyclone Interchange to work together. For detailed product information, refer to the respective product's documentation.

What You Need to Know

This document is written for users who have experience using WebLogic Integration and Cyclone Interchange, or who have received training in both products. In addition, users should have a basic understanding of Java Message Service (JMS), Web services, and XML. If you are planning to use WebLogic Integration and Cyclone Interchange in a cluster, then a general understanding of clustering is recommended. If you are going to be trading ebXML documents, knowledge of ebXML and Collaboration Protocol Agreements (CPAs) is required.

Product Documentation on the dev2dev Web Site

BEA product documentation, along with other information about BEA software, is available from the BEA dev2dev Web site:

`http://dev2dev.bea.com`

To view the documentation for a particular product, select that product from the list on the dev2dev page; the home page for the specified product is displayed. From the menu on the left

side of the screen, select Documentation for the appropriate release. The home page for the complete documentation set for the product and release you have selected is displayed.

Related Information

Readers of this document may find the following documentation and resources especially useful:

- For general information about Java applications, go to the Sun Microsystems, Inc. Java Web site at `http://java.sun.com`.
- For general information about XML, go to the O'Reilly & Associates, Inc. `XML.com` Web site at `http://www.xml.com`.
- For information about Cyclone Exchange, see the Cyclone Commerce Web site at `http://www.cyclonecommerce.com/`.

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Integration **Product Version:** .

If you have any questions about this version of BEA WebLogic Integration, or if you have problems installing and running BEA WebLogic Integration, contact BEA Customer Support at **http://support.bea.com**. You can also contact Customer Support by using the contact information provided on the quick reference sheet titled "BEA Customer Support," which is included in the product package.

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.
`monospace text`	Indicates *user input*, as shown in the following examples: • Filenames: `config.xml` • Pathnames: *`BEAHOME`*`/config/examples` • Commands: `java -Dbea.home=BEA_HOME` • Code: `public TextMsg createTextMsg(`
	Indicates *computer output*, such as error messages, as shown in the following example: `Exception occurred during event dispatching:java.lang.ArrayIndexOutOfBoundsException: No such child: 0`
`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
`{ }`	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*: `java utils.MulticastTest -n` *`name`* `[-p` *`portnumber`*`]`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed. *Example*: `java weblogic.deploy [list|deploy|update]`

Convention	Item
. . .	Indicates one of the following in a command line: • That an argument can be repeated several times in a command line • That the statement omits additional optional arguments • That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name] [-f "file1.cpp file2.cpp file3.cpp . . ."`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Integrating BEA WebLogic Integration and Cyclone Interchange

This document describes various options for trading ebXML and EDI documents among B2B collaborators using WebLogic Integration (version 8.1.4; this solution has also been certified with versions 8.1.5 and 8.5) and Cyclone Interchange (version 5.3.2) in an integrated environment. This document describes a range of common trading scenarios, from single end-point solutions to advanced clustered configurations that offer fault tolerance and high availability. This document focuses on ways to configure WebLogic Integration and Cyclone Interchange to work together.

Note: This solution has been certified for WebLogic Integration versions 8.1.4, 8.1.5, and 8.5; and for Cyclone Interchange version 5.3.2.

This document contains the following sections:

Product Introduction

This section introduces BEA WebLogic Integration and Cyclone Interchange. For detailed product information, refer to the respective product's documentation.

About BEA WebLogic Integration

BEA WebLogic Integration provides customers with a unified framework for business integration, simplified production and management, as well as a new extensible architecture for the rapid assembly and integration of applications, business processes and partner trading communities. By leveraging the BEA WebLogic Workshop development environment, pre-built and custom controls, and a catalog of standards-based application adapters, WebLogic Integration presents a seamless production environment for integration specialists and application developers alike, providing a unified platform to build, extend, integrate, deploy and manage applications as end-to-end business processes. For general information about WebLogic Integration, see the BEA Systems, Inc. web site at http://www.bea.com. For WebLogic Integration product documentation, see http://dev2dev.bea.com/wlintegration/.

About Cyclone Interchange

Cyclone Interchange is a B2B e-commerce gateway that provides a secure, scalable foundation for B2B or internal collaboration. The unified framework helps establish relationships with trading partners, transact business over the Internet, and integrate with back-end systems. Cyclone Interchange provides flexibility in connecting with partners and legacy systems using widely used protocols, transports, and integration methods. For Cyclone Interchange product information, see http://www.cyclonecommerce.com/products/interchange.php.

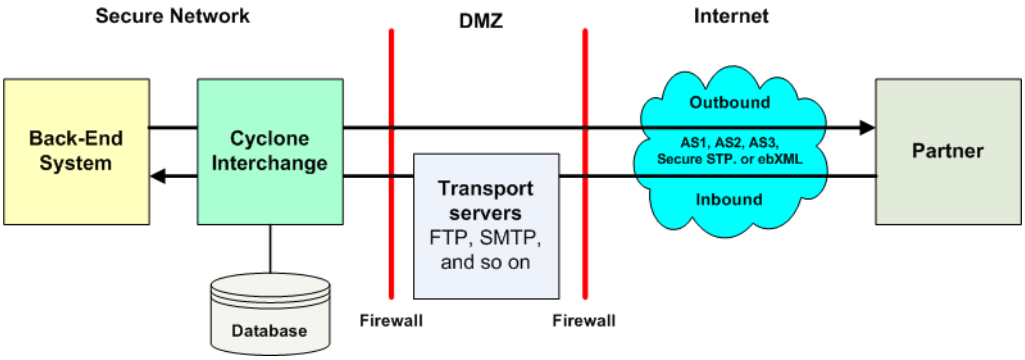
Cyclone Interchange can be implemented as a single end-point solution or as a clustered, fault-tolerant gateway with unlimited trading partners. Cyclone Interchange supports standard protocols for exchanging messages with partners, including AS1, AS2, AS3, ebXML, and RosettaNet. User-defined message routing and rules-based processing can be handled at the partner or document level.

Cyclone Interchange supports multiple operating systems, databases, and platforms. For more information about supported products, refer to the "System Requirements" chapter of the Cyclone Interchange *Installation and Configuration* document.

Cyclone Interchange has been certified for AS1, AS2, AS3, and ebXML interoperability. For a list of software for which Cyclone Interchange has been interoperability certified, see the eBusinessReady Web site at http://www.ebusinessready.org/.

Figure 1 shows a typical architecture for Cyclone Interchange.

Figure 1 Typical Architecture for Cyclone Interchange



Cyclone Interchange also supports other architectures, but this example is popular among business users. The exact architecture for a specific organization depends on many factors, including that organization's security policies, preferred protocols for exchanging documents over the Internet, back-end integration requirements, and other requirements.

Cyclone Interchange's user interface integrates gateway management, monitoring, and metrics into a single view. Figure 2 shows a portion of the Cyclone Interchange user interface. Hyperlinked image components help users visualize and manage the trading configuration.

Figure 2 Hyperlinked Trading Visualization from the Cyclone Interchange User Interface

Terminology Used in This Document

The following terms are used in this document.

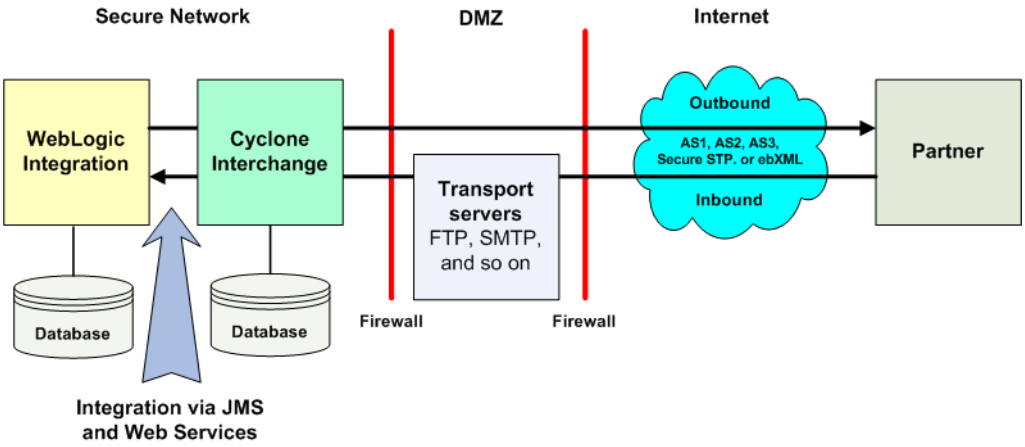
Term	Definition
B2B	Business-to-business electronic commerce. B2B covers a broad range of business activities. For example, B2B systems exchange business documents, such as purchase orders and invoices, between partners in a supply chain.
back-end system	A system (such as WebLogic Integration) that sends documents to Cyclone Interchange to be delivered across the Internet, or to which Cyclone Interchange sends documents it receives across the Internet.
control	Component that can communicate with other applications and components. For example, a database control lets a Web service request data from a database.
document	Message traded between partners. In this document, a business document is assumed to be an EDI or ebXML document.
inbound	Message received over the Internet from a partner. A trading engine unpacks an inbound message and sends it to a back-end system for processing.
Integration Delivery Exchange	Transport for delivering inbound messages (that Cyclone Interchange receives from a trading partner) to a back-end system. Supported integration protocols include file system, FTP, JMS, Web services and MQ Series.
Integration Pickup Exchange	Transport for picking up outbound messages from a back-end system. Cyclone Interchange sends outbound messages a trading partner. Supported integration transports include file system, FTP, JMS, Web services and MQ Series.
message	Business document traded between partners. In this document, a business document is assumed to be an EDI or ebXML document.
outbound	Message sent over the Internet to a partner. The trading engine picks up the outbound message from a back-end system and packages it before sending to the partner.
trading	Exchange of e-commerce messages over the Internet between trading partners.
trading engine	B2B e-commerce gateway, such as Cyclone Interchange.

Term	Definition
WLI Control	Java class that facilitates access to an enterprise resource. For more information, see the WebLogic Integration documentation.
WLI Event Generator	Publish messages to WebLogic Integration Message Broker channels in response to system events. For more information, see the WebLogic Integration documentation.

Integrated B2B Solution

In combination, WebLogic Integration and Cyclone Interchange together provide an *integrated B2B solution* that supports trading ebXML, EDI X12, and EDIFACT documents. Figure 3 provides an architecture overview of this integrated B2B solution:

Figure 3 B2B Integration Architecture for WebLogic Integration and Cyclone Interchange



This tested solution leverages the strengths of WebLogic Integration's workflow engine and the proven reliability of Cyclone Interchange's standards-based B2B trading. Integration solutions rely on either JMS or Web services as the integration mechanism between the two products. WebLogic Integration and Cyclone Interchange can run on the same server, or they can be deployed on separate, independent servers. Both applications can be clustered.

This section describes the following options for the integrated solution:

- JMS Integration
- JMS Clustering

- Web Services Integration
- Web Services Clustering

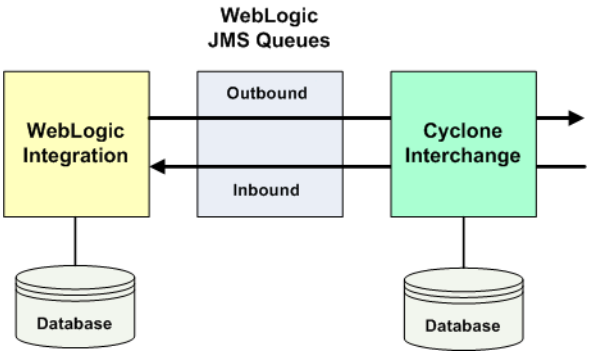
JMS is the recommended transport mechanism for exchanging messages up to 4MB in size. Web services is the recommended transport mechanism for exchanging messages larger than 4MB.

JMS Integration

Java Message Service (JMS) offers a reliable integration mechanism between WebLogic Integration's Business Process Management and Cyclone Interchange. JMS is the recommended transport mechanism for exchanging messages up to 4MB in size.

As shown in Figure 4, the messaging service provides a layer of separation between the applications, which means that the underlying integration transport is reliable and independent of both WebLogic Integration and Cyclone Interchange.

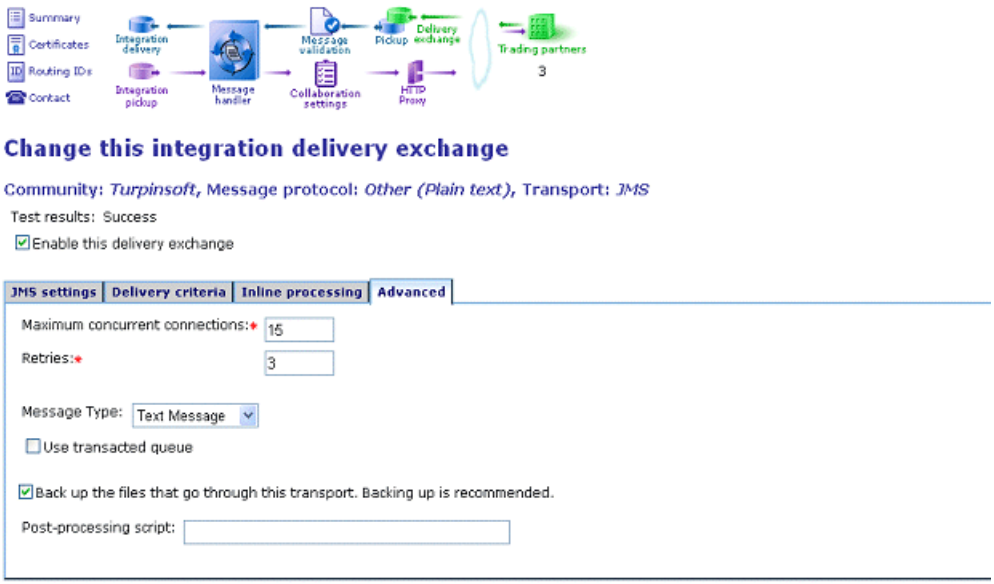
Figure 4 JMS Integration



JMS queues can be configured in WebLogic Server. Persistent queues are recommended to preserve messages in the event of a server outage. Persistent queues should be configured first in the WebLogic Server Console, and then subsequently in Cyclone Interchange.

In Cyclone Interchange, two JMS delivery exchanges are required: one for integration pickup and the other for integration delivery. While configuring the JMS delivery exchange, administrators should select the correct JMS message type (BytesMessage or TextMessage). This decision should be based on what type the WebLogic Integration business process's message broker channel is expecting. The JMS message type can be configured on the Advanced tab of the JMS delivery exchange's maintenance page in the Cyclone Interchange user interface, as shown in the following example.

Figure 5 JMS Integration Delivery Exchange Settings



There are two main scenarios to configure, which are based on the message direction:

- **Outbound scenario** (WebLogic Integration sends messages to Cyclone Interchange)—WebLogic Integration uses the JMS Control to send JMS messages to the JMS Queue.
- **Inbound scenario** (WebLogic Integration receives messages from Cyclone Interchange)—The JMS Event Generator needs to be configured in the WebLogic Integration Console to map the physical JMS queue to a message broker channel. WebLogic Integration business processes then use this channel to listen for the incoming messages from Cyclone Interchange. For more information, see "Deploying Event Generators" in "Understanding WebLogic Integration Clusters" in *Deploying WebLogic Integration Solutions* at:

  `http://e-docs.bea.com/wli/docs81/deploy/cluster.html`

When configuring Cyclone Interchange for JMS, the weblogic.jar file must be included in the classpath of the Cyclone Interchange server by placing a copy in the following location:

`[install directory]\[build number]\corelib`

For details about JMS configuration, refer to the "JMS Transport" chapter in the "Delivery Exchanges" chapter of the Cyclone Interchange *Installation and Configuration Guide*. Note that the Cyclone Interchange engine should be restarted after modifying the classpath.

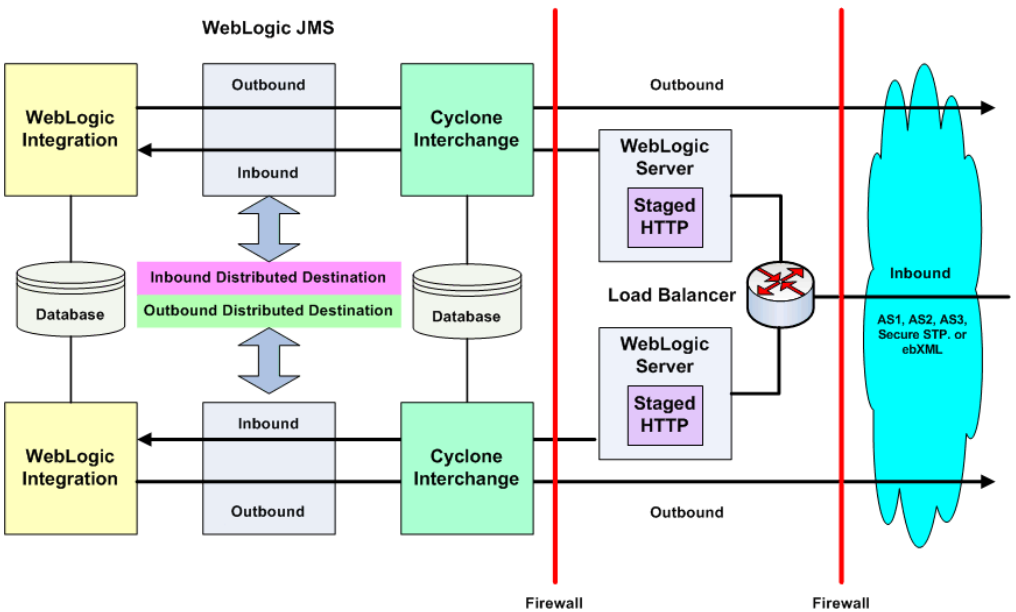
JMS Clustering

If WebLogic Integration and Cyclone Interchange are deployed in a clustered environment, then JMS queues must be clustered in WebLogic Server via the use of distributed destinations. A distributed destination is a set of physical destinations (queues or topics) that are called under a single JNDI name so that they appear to be a single, logical destination to a client, even when the members of the set are actually distributed across multiple servers within a cluster, with each destination member belonging to a separate JMS server. Distributed destinations rely on physical JMS queues running on the servers in the cluster. For more information, see "Using Distributed Destinations" in *Programming WebLogic JMS* the WebLogic Server documentation at:

http://e-docs.bea.com/wls/docs90/jms/dds.html

Figure 6 shows WebLogic Integration and Cyclone Interchange using distributed destinations, providing a clustered layer between both applications.

Figure 6 Distributed Destinations in a Clustered Environment



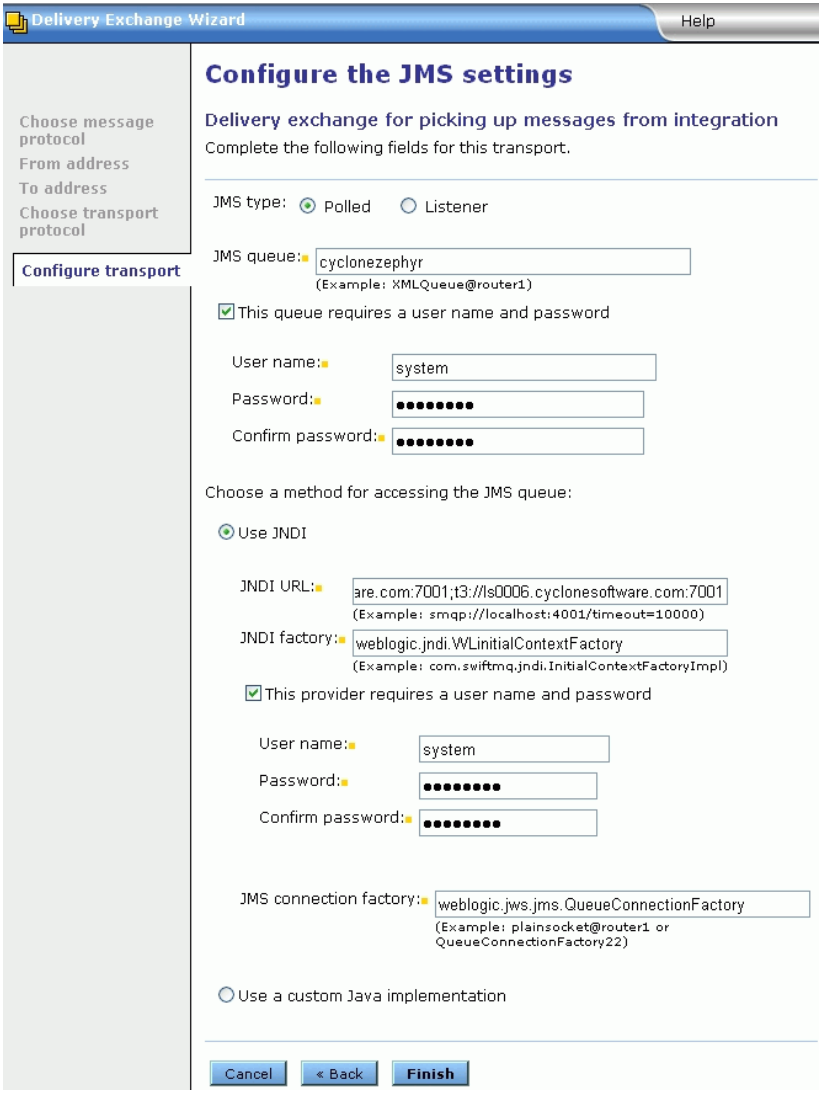
Note: This figure shows Cyclone Interchange's staged HTTP servlet running in a high availability configuration on WebLogic Web servers in the DMZ to handle inbound messages. For more information, see "High Availability Staged HTTP" on page 18.

When configuring Cyclone Interchange, the only difference between using a physical JMS queue and a distributed destination is the JNDI URL. The JNDI URL should be a semi-colon-separated list of the servers that are part of the distributed destination, as shown in the following example (note that a semicolon separates the URLs of each server):

```
t3://ls0005.cyclonesoftware.com:7001;t3://ls0006.cyclonesoftware.com:7001
```

Figure 7 shows the Configure the JMS Settings page in the Cyclone Interchange user interface, which is used to set up a JMS integration pickup exchange.

Figure 7 JMS Transport Page in Cyclone Interchange

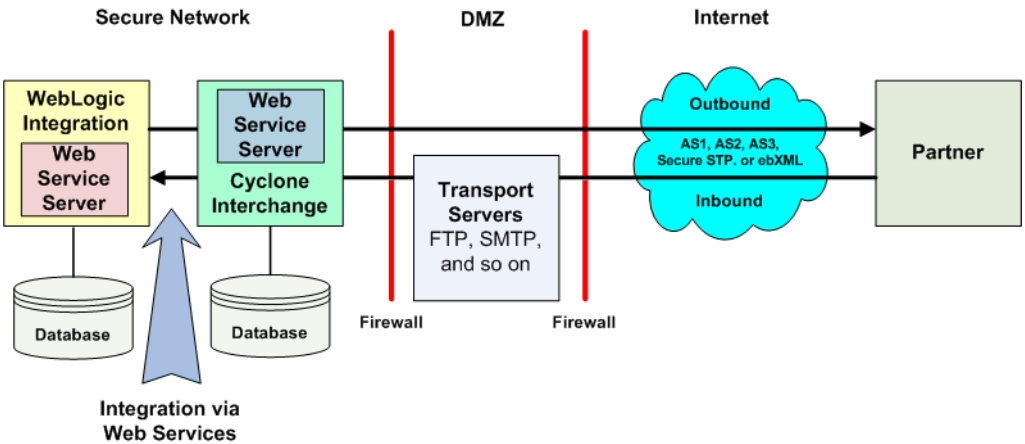


Web Services Integration

Web services provide another integration option for WebLogic Integration and Cyclone Interchange. Web services is the recommended transport mechanism for exchanging messages larger than 4MB.

Figure 8 shows Web services integration for WebLogic Integration and Cyclone Interchange.

Figure 8 Web Services Integration



Cyclone Interchange includes a built-in Web service that exposes a standard interface (API) in the form of a WSDL (MessageService.wsdl). This file resides in the following directory:

`[install directory]\[build number]\conf`

Outbound Scenario

For the outbound scenario, clients must invoke operations on the MessageService.wsdl in order to send documents to the Cyclone Interchange's Web service. In the outbound scenario, WebLogic Integration uses a Web Service control to communicate with the Cyclone Interchange's Web service. The Web Service control must implement the MessageService.wsdl so that the workflow can invoke the methods exposed by the WSDL with the help of the Web Service control.

To send the ebXML/EDI documents, the Web Service control can use either of two methods (`submitMessage` or `submitPayload`) exposed by the WSDL. To configure outbound communication:

1. Implement the `submitMessage` or `submitPayload` operation to allow workflows to send a single outbound document to Cyclone Interchange.
2. Change the URL in the MessageService.wsdl to include the server and port for the Cyclone Interchange server or load balancer IP.

   ```
   <wsdlsoap:address
   location="http://localhost:5080/services/MessageService"/>
   ```

3. After modifying the Cyclone Interchange location, always rebuild the WebLogic Integration application.

Although there are no file size limits with Web Services integration, consider the impact of document size at run time. If your organization plans to trade large files over Web services, see "Trading Large Files" on page 19.

When designing a WebLogic Integration business process to submit messages to the Cyclone Interchange Web service, remember to implement retry logic in the event that the Web Service control is unable to communicate with the Cyclone Interchange Web service. Cyclone Interchange may be unavailable because it is shut down (planned or unplanned), or because it is temporarily overloaded. Cyclone Interchange throws a SOAP fault if it is too busy to handle the Web service request, as shown in the following example.

Listing 1 Cyclone Interchange Overburden Fault in a Web Service Control

```
<?xml version="1.0" encoding="UTF-8"?>

<xml-fragment xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns2="http://xml.apache.org/axis/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="http://cyclonecommerce.com/tradingengine/transport/api/webservi
ce" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

   <faultcode>soapenv:Server.userException</faultcode>

   <faultstring>java.lang.Exception: Failed: System capacity has been
exceeded</faultstring>

   <detail>

      <ns1:fault
xmlns:ns1="http://cyclonecommerce.com/tradingengine/transport/api/webservi
ce"/>
```

```
      <ns2:hostname
xmlns:ns2="http://xml.apache.org/axis/">ls0006</ns2:hostname>
   </detail>
</xml-fragment>
```

Inbound Scenario

For the inbound scenario, in order to receive documents from Cyclone Interchange, WebLogic Integration must:

- expose a Web service that implements MessageService.wsdl (which resides in `[install directory]\[build number]\conf`)
- provide an implementation (containing the business logic) for the `sendMessage` method

Cyclone Interchange always invokes the `sendMessage` method to send ebXML/EDI documents to WebLogic Integration. If the back-end system became unavailable for some reason, Cyclone Interchange would automatically retry three times (or the number of times configured by the user), and then, if retries are exhausted, mark the message as 'Failed' in its user interface. The administrator can manually resubmit the failed messages when the back-end system becomes available.

In Cyclone Interchange, two Web services delivery exchanges are required: integration pickup and integration delivery. For more information, see the “Web Services API Transport” topic in the “Delivery Exchanges” chapter of the Cyclone Interchange *Installation and Configuration Guide*.

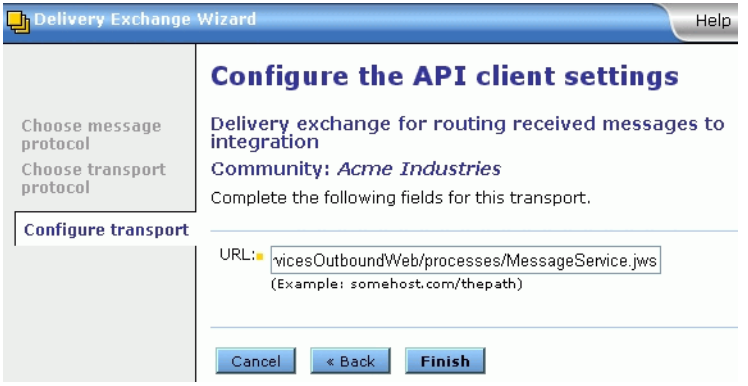
In Cyclone Interchange, when configuring integration delivery for delivering messages to WebLogic Integration, the URL should be that of the service endpoint URI. For example, for a Web service implemented in WebLogic Integration, the URI will be similar to the following example:

```
http://server.host.com:7001/CycloneB2BWebServicesOutboundWeb/processes/MessageService.jws
```

In this example, MessageService.jws provides the implementation for the interface described in the MessageService.wsdl.

Figure 9 shows the Configure the API client settings page in the Cyclone Interchange user interface, which is where Web services integration delivery exchange is configured.

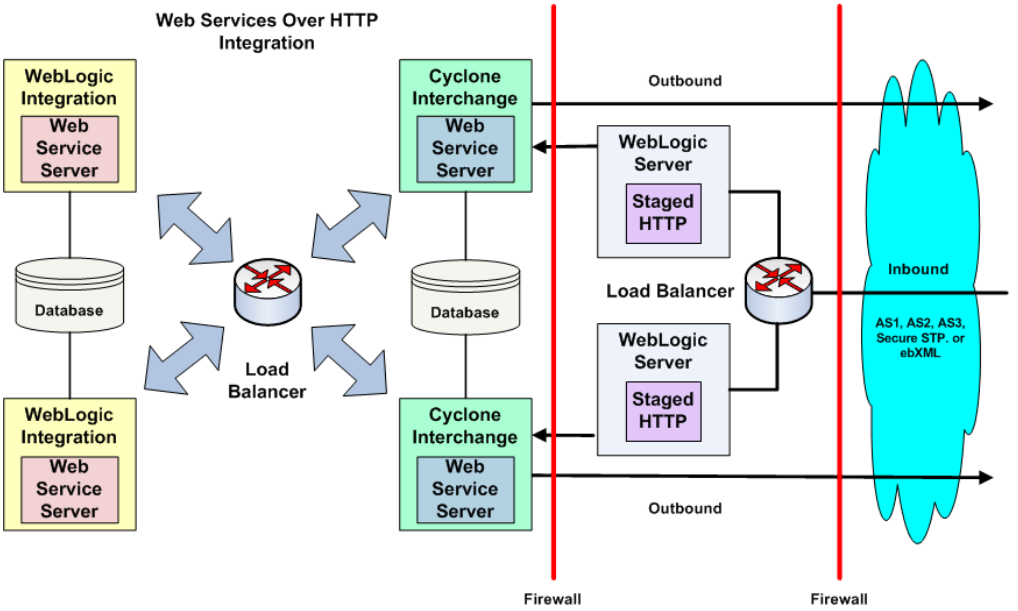
Figure 9 Web Services Transport Page in Cyclone Interchange



Web Services Clustering

HTTP load balancing is required in a clustered environment of WebLogic Integration and Cyclone Interchange. The load balancer redirects message traffic when one service in the cluster becomes unavailable. Figure 10 illustrates Web services clustered integration for WebLogic Integration and Cyclone Integration.

Figure 10 Web Services Clustered Integration With Load Balancer



For inbound traffic from Cyclone Interchange, WebLogic Integration provides a software load balancer that can be configured when creating a clustered domain. For outbound traffic to Cyclone Interchange, either a software- or hardware-based load balancer is required. For outbound traffic from WebLogic Integration to Cyclone Interchange, load balancer hardware is required. The alternative is to use load balancer hardware for traffic in both directions.

WebLogic Integration and Cyclone Interchange must be configured to use the virtual IP address of the load balancer.

- **WebLogic Integration**—For outbound traffic to Cyclone Exchange, for the Web Service control, change the URL of the service in the MessageService.wsdl to the virtual IP address by changing the following element:

  ```
  '<wsdlsoap:address
  location="http://localhost:5080/services/MessageService"/>"'
  ```

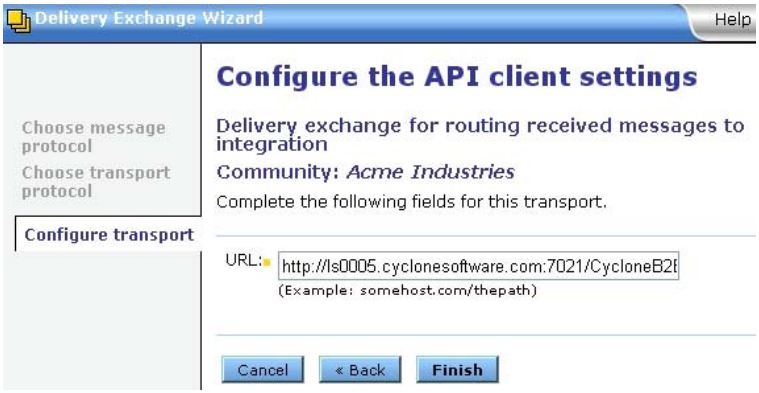
- **Cyclone Interchange**—For inbound traffic from Cyclone Exchange, change the Web Service's exchange URL to include the virtual IP, as shown in the following example.

  ```
  http://ls0005.cyclonesoftware.com:7021/CycloneB2BWebServicesOutboundWeb
  /processes/MessageService.jws
  ```

In this example, the WebLogic load balancer application runs on port `7021` of the server named `ls0005`.

Figure 11 shows the Configure the API client settings page in the Cyclone interchange user interface, which is where the URL is configured.

Figure 11 Web Services Clustering URL Example



Integration Guidelines for ebXML Documents

Setting up a deployment that involves trading documents over ebXML requires prior knowledge of ebXML standards and Collaboration Protocol Agreements (CPAs). The CPA document controls messaging characteristics, such as whether acknowledgements are used, signing and encryption, and duplicate elimination.

Specifying Metadata

When WebLogic Integration sends an ebXML document to Cyclone Interchange, WebLogic Integration must provide specific metadata.

- For JMS integration, the metadata is defined in the JMS message properties.
- For Web services integration, the metadata is defined in the WSDL.

This metadata is required to integrate the ebXML Message Handler (Cyclone Interchange) with the backend system, per the relationship between the CPA and the BPSS (Business Process Specification Schema) documents.

Monitoring Delivery Success / Failure

To keep track of the delivery success/failure of an outbound payload that it sends to Cyclone Interchange, WebLogic Integration can set the `IntegrationId` metadata for the outbound message. To do this, WebLogic Integration must first register to receive event notifications from Cyclone Interchange (see the "Events System" chapter in the Cyclone Interchange *Installation and Configuration* document for instructions). Once configured, Cyclone Interchange will return an event notification containing an `IntegrationId` attribute that corresponds to the outbound payload. For example, suppose WebLogic Integration sends two payloads to Cyclone Interchange, one bound for partner A (`IntegrationId=1`) and another bound for partner B (`IntegrationId=2`). Suppose that the first payload is delivered successfully but that delivery of the second payload fails after several retries because partner B is down. In this scenario, Cyclone Interchange will fire two event notifications to WebLogic Integration that communicate something to the effect of:

- `"message delivered, IntegrationId=1"`
- `"message failed, IntegrationId=2"`

Note: The `IntegrationId` mechanisms works for both JMS and Web services integration.

Correlating Responses With Requests

WebLogic Integration can use the `RefToMessageId` metadata element to correlate a business response with its business request. For an outgoing document, specify this metadata element in the message. To link an outbound document to an inbound document, this metadata element must contain the same value. For example, a 3A4 (Purchase Order) is a business request and a 3A7 (Purchase Order Acknowledgement) is a business response. When Cyclone Interchange delivers the 3A4 to WebLogic Integration, the `MessageId` field will be populated with a unique value (for example, `123`). When WebLogic Integration responds by providing a 3A7 document to Cyclone Interchange, WebLogic Integration needs to populate the `RefToMessageId` metadata with the corresponding value (in our example, `123`). This value is what Cyclone Interchange and the trading partner will use to link the specific request (3A4) with the specific response (3A7).

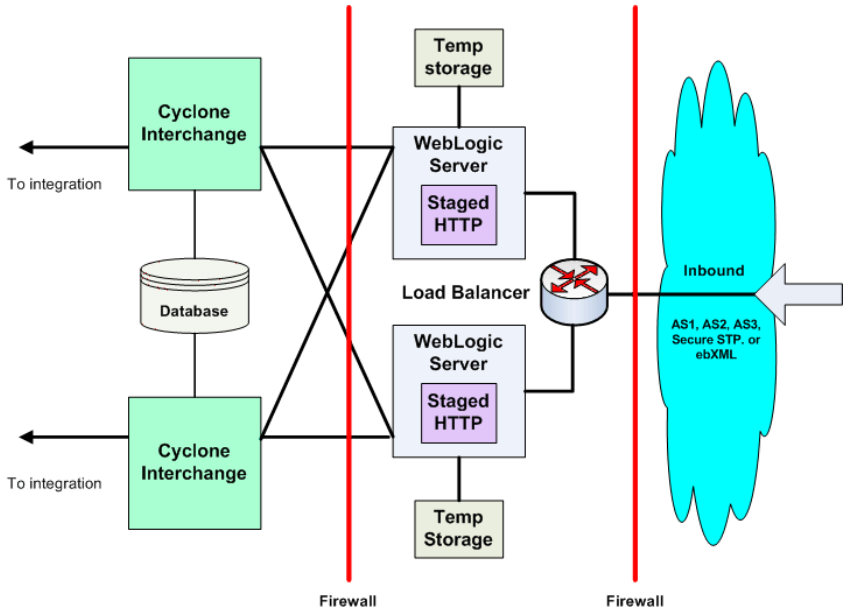
Further Reading

Users implementing controls for WebLogic Integration would benefit from learning about the Cyclone Interchange message metadata required for ebXML. For more information, refer to the "ebXML Support" chapter of the Cyclone Interchange *Installation and Configuration* documentation.

High Availability Staged HTTP

Figure 12 shows Cyclone Interchange deployed in a high availability configuration to handle inbound traffic.

Figure 12 Staged HTTP High Availability Configuration



Note the multiple instances of the Cyclone Interchange Staged HTTP servlet configured in a cluster. This approach involves configuring Staged HTTP servlets on multiple WebLogic Web servers. A load balancer (which can be either software- or hardware-based) directs inbound traffic to each Web server and handles automatic failover in the event that an instance of the Staged HTTP servlet fails. A trading partner sends messages using the virtual IP address of the load balancer. In this clustered configuration, each Cyclone Interchange instance requires a staged HTTP delivery exchange (the message protocol that a community or partner uses to send and receive messages) for receiving messages from trading partners.

For instructions on setting up the staged HTTP servlet, refer to the "Staged HTTP" chapter of the Cyclone Interchange *Installation and Configuration* document. To configure for high availability, follow the same instructions for multiple Web servers.

Trading Large Files

Various affect how your organization trades with partners, including:

- your organization’s infrastructure, standards, and policies
- your organization’s hardware and software, preferred trading protocols, preferred message transports, and security policies
- your partners' business practices

Such factors might require your implementation to handle the exchange of large files among trading partners.

Which Protocol to Use—AS1, AS2, or AS3?

Other factors aside, AS3 FTP and AS2 HTTP are better suited for trading large documents between partners than AS1 SMTP (or ebXML SMTP). The latter can be less desirable due to widespread corporate policies that impose limits on the size of e-mail messages.

When choosing trading protocols, you and your partners should consider factors that may affect file size. For example:

- For AS1 SMTP, know the message size limitations for mail servers in your organization as well as in your partners' organizations.
- For AS2 HTTP and ebXML HTTP, set connection time-outs sufficiently high to allow documents to be traded successfully. Also, for large documents, asynchronous receipts are preferred, as synchronous responses can hold HTTP connections open for too long.

File Sizes in WebLogic JMS Integration

JMS is the recommended transport mechanism for exchanging messages up to about 4MB in size.

File Sizes in Web Services Integration

Web services is the recommended transport mechanism for exchanging messages larger than 4MB. For extremely large documents (larger than approximately 250 megabytes), use file system integration instead, as described in “File System Integration” on page 21.

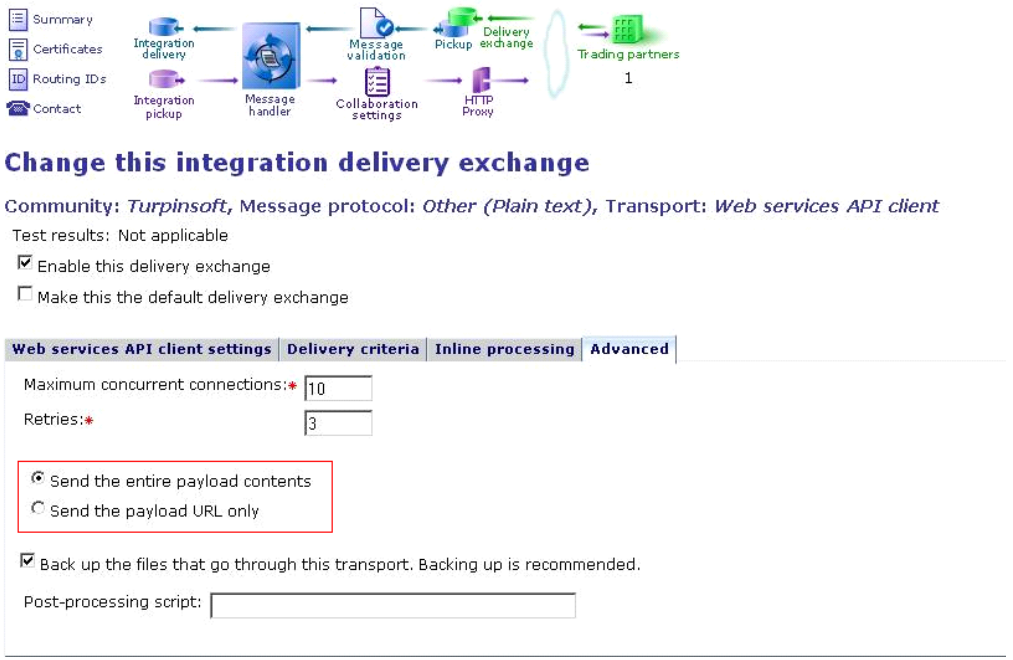
The Cyclone Interchange Web Services API supports two ways of moving documents between Cyclone Interchange and WebLogic Integration:

- Passing entire documents over HTTP. For large documents, this approach is resource-intensive, and network time-outs might occur.
- Send an URL that references the document rather than sending the document itself. The URL points to the document on a file system shared by WebLogic Integration and Cyclone Interchange. This approach reduces network traffic and works well for documents of up to approximately 250 megabytes in size. WebLogic Integration and Cyclone Interchange can run on the same server or reside on different servers. Because the syntax of the path in the URL differs on UNIX and Windows, this approach requires that WebLogic Integration and Cyclone Interchange both run on the same operating system—both on UNIX or both on Windows.

To configure Cyclone Interchange to send the URL but not the document:

1. Open the Integration Delivery Exchange for Web Services page in the Cyclone Interchange user interface.
2. On the Advanced tab, select Send the payload URL only.

Figure 13 Integration Delivery Exchange Page



3. If you are using a Web Service control to send outbound documents using the URL, set the URL in the **URL** field rather than the **Content** field of the payload (because the payload can contain either content or an URL).

For this approach, the approximate 250MB file size limit is due to the way that Cyclone Interchange backs up documents. When a document is received from the Web Services API, it must be backed up before the message ID can be returned to WebLogic Integration. For large documents, the Cyclone Interchange backup operation can require considerable time. During backup, the HTTP connection between WebLogic Integration and Cyclone Interchange remains open until Cyclone Interchange returns the message ID, so connection time-outs might occur for large documents. The actual maximum file size will depend on the speed with which Cyclone Interchange can back up documents, as well as the time-out value used by the back-end client system.

File System Integration

For documents larger than approximately 250MB, use file system integration for WebLogic Integration and Cyclone Interchange. For configuration instructions, refer to the "Delivery Exchanges" chapter in the Cyclone Interchange *Installation and Configuration* document.

Disk Volume Guidelines

For deployments in which large files will be traded, capacity planning for storage space requirements is critical. Requirements analysis should include an estimate of the maximum volume of documents that might be traded at one time, and should consider growth in document volume over time. Cyclone Interchange uses a temporary directory to store documents in process. The available space of the temporary directory should be at least three times larger than the maximum aggregate size of the documents being processed.

The path of the Cyclone Interchange temporary directory is defined in the filereg.xml file, which resides in the following location:

`[install directory]\[build number]\conf`

Tuning Guidelines for WebLogic Integration

Performance tuning becomes more critical for deployments in which large files are traded. For information about performance tuning in WebLogic Integration, see:

http://e-docs.bea.com/wls/docs81/perform/index.html

For information about performance tuning in Cyclone Interchange, refer to the Cyclone Interchange *Installation and Configuration* document.

Index

T

W