BEA WebLogic Collaborate

A Component of BEA WebLogic Integration

Implementing RosettaNet for BEA WebLogic Collaborate

BEA WebLogic Collaborate Release 2.0
Document Edition 2.0
July 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Implementing RosettaNet for BEA WebLogic Collaborate

Document Edition	Date	Software Version
2.0	July 2001	2.0

Contents

3. Using Workflows with RosettaNet

4. RosettaNet PIP Templates

Index

About This Document

This document describes how to use RosettaNet in a BEA WebLogic Collaborate™ environment.

BEA WebLogic Collaborate introduces a routing architecture that allows it to manage and resolve XOCP, RosettaNet, and cXML messages. This architecture allows trading partners using BEA WebLogic Collaborate to engage in business-to-business conversations using any of these protocol standards.

RosettaNet on BEA WebLogic Collaborate provides the ability to send and receive RosettaNet messages as described in *RosettaNet Implementation Framework 1.1* (RNIF 1.1) and *RosettaNet Implementation Framework 2.0* (RNIF 2.0). In addition, integrated extensions to WebLogic Process Integrator support the modeling, creation, and execution of workflows that model RosettaNet Partner Interface Processes (PIPs). RosettaNet on BEA WebLogic Collaborate includes two PIP templates (0A1 and 3A2), which you can use as the basis for your own PIP implementations. Samples have also been developed, and are available at the BEA Developer's Center.

This document is organized as follows:

- Chapter 1, "Introduction," provides an introduction to RosettaNet on BEA WebLogic Collaborate, the RosettaNet Implementation Framework 1.1 and 2.0 (RNIF 1.1 and RNIF 2.0), and the architecture used to implement RosettaNet on BEA WebLogic Collaborate.
- Chapter 2, "Configuring RosettaNet Security," describes security issues for RosettaNet implementations on BEA WebLogic Collaborate.
- Chapter 3, "Using Workflows with RosettaNet," describes how to use the WebLogic Process Integrator Studio to create workflows for use with RosettaNet.

- Chapter 4, “RosettaNet PIP Templates,” describes the two RosettaNet PIP templates provided with BEA WebLogic Collaborate, and explains how to create more PIPs of your own.

What You Need to Know

This document is intended primarily for:

- Business process designers who use the WebLogic Process Integrator Studio to design workflows that can be integrated with the WebLogic Collaborate environment, specifically focusing on RosettaNet implementations.
- System administrators who set up and administer WebLogic Collaborate applications in a RosettaNet environment.

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/.

Related Information

For more information about Java 2 Enterprise Edition (J2EE), Extended Markup Language (XML), and Java programming, see the *BEA WebLogic Collaborate Glossary* in the WebLogic Collaborate online documentation.You can also refer to the Javasoft Web site:

`http://java.sun.com`

For more information about RosettaNet, visit the RosettaNet Consortium Web site:

`http://www.rosettanet.org`

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`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR

Convention	Item
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

1 Introduction

The following sections provide an overview of the RosettaNet protocol:

- About RosettaNet
- RosettaNet Architecture
- RosettaNet Administration

About RosettaNet

This section introduces the RosettaNet standard for electronic business transactions. The RosettaNet Consortium is an independent, nonprofit consortium of major information technology, electronic component, and semiconductor manufacturing companies working to create and implement industry-wide, open e-business process standards. These processes are designed to standardize the electronic business interfaces used between participating supply chain partners. The *RosettaNet Implementation Framework* specification (available at `http://www.rosettanet.org`) is a guideline for applications that implement RosettaNet *Partner Interface Processes* (PIPs). These PIPs are standardized electronic business processes used between trading partners.

RosettaNet Architecture

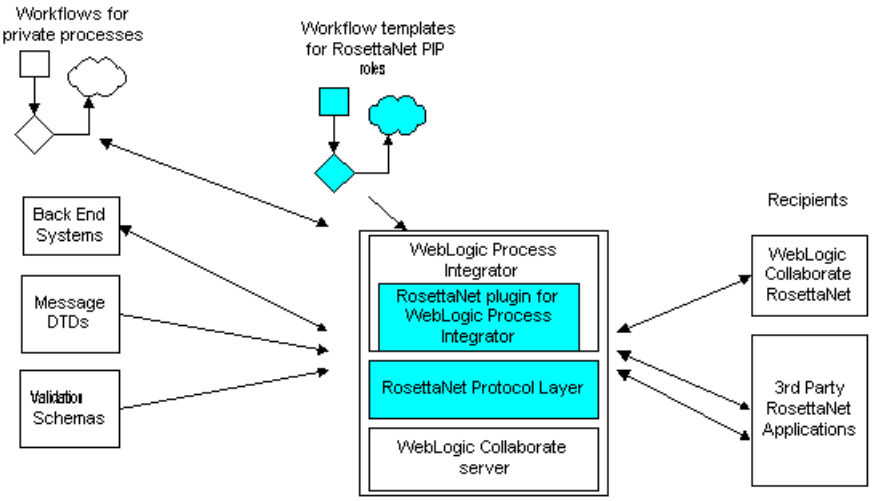
BEA WebLogic Collaborate support for RosettaNet consists of the following components:

- RosettaNet protocol layer
- WebLogic Collaborate RNIF 1.1 and RNIF 2.0 plug-ins for WebLogic Process Integrator, providing support for workflow modeling and execution of RosettaNet PIPs
- PIP templates

WebLogic Collaborate Architecture and RosettaNet

The following diagram shows how WebLogic Collaborate supports the RosettaNet architecture, and how WebLogic Collaborate interacts with other systems using RosettaNet.

Figure 1-1 WebLogic Collaborate RosettaNet Architecture



WebLogic Collaborate support for RosettaNet is designed to integrate RosettaNet seamlessly with the standard WebLogic Collaborate infrastructure. For more information about the remainder of the WebLogic Collaborate architecture, see *Introducing BEA WebLogic Collaborate*.

RosettaNet Protocol Layer

The RosettaNet protocol layer provides the ability to send and receive messages by way of the Internet according to the RNIF 1.1 and RNIF 2.0 specifications for transport, message packaging, and security.

PIP Templates

RosettaNet PIPs define the *public processes* in which trading partners participate while performing e-business transactions. For example, PIP 3A2 defines the process that a *Customer* trading partner performs with a *Product Supplier* trading partner to get information about the price and availability of goods that the *Customer* wants to buy and the *Product Supplier* wants to sell. Trading partners participating in PIPs need to implement the public process defined by their role in the PIP, and they need to connect their internal systems, as well as their private processes and workflows, to the public process.

A key feature of the support for RosettaNet provided by WebLogic Collaborate is the set of WebLogic Process Integrator workflow PIP templates that trading partners can use to implement PIPs. Template sets are defined for the following PIPs:

- PIP0A1: Notification of Failure
- PIP3A2: Query Price and Availability

For further information about these and other PIPs, go to the RosettaNet Web site: `http://www.rosettanet.org`. For more information on the RosettaNet PIPs provided with this release of BEA WebLogic Collaborate, see Chapter 4, “RosettaNet PIP Templates.”

Integration

BEA WebLogic Process Integrator models and executes workflows that implement RosettaNet PIPs. The RosettaNet protocol layer and WebLogic Process Integrator are integrated to provide workflows with the following capabilities:

- WebLogic Process Integrator actions to send RosettaNet messages
- WebLogic Process Integrator events to wait for RosettaNet messages
- Workflow template properties to indicate which PIP and role the workflow implements
- The ability to pass incoming RosettaNet messages to the correct workflow instance and to initiate the correct type of workflow upon receipt of an initiating RosettaNet message

Workflow Definition

To support the development of workflows for RosettaNet PIPs, WebLogic Collaborate provides the same tools used for modeling other WebLogic Collaborate workflow processes. All RosettaNet workflows use a common interface, based on WebLogic Process Integrator, to define workflow actions, events, and logic flows.

Digital Signatures

WebLogic Collaborate includes an out-of-the-box implementation of digital signatures, based on the RSA CertJ toolkit. For more information about implementing and configuring digital signatures, see *Using BEA WebLogic Collaborate Security*..

Message Validation

The RosettaNet PIP definitions contain detailed validation rules for messages exchanged in the PIP. These rules are significantly more stringent than the validation expressed within an XML Document Type Definition (DTD).

The required validation rules are expressed in XML schema documents (XSD), which are included with the PIP templates provided with BEA WebLogic Collaborate. For a more detailed description of message validation, see Chapter 3, "Using Workflows with RosettaNet."

Samples

BEA has developed several RosettaNet samples to demonstrate how RosettaNet functionality operates under WebLogic Collaborate. For more information on these samples, please visit the BEA Developer Center Web site:

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

Unsupported Items

Several RosettaNet-related features are not supported in this release of BEA WebLogic Collaborate:

- SMTP Transport: RNIF 2.0, while strongly biased to HTTP transport, is transport independent; it includes documentation showing how SMTP transport might be used. BEA WebLogic Collaborate supports HTTP and HTTPS transport; it does not support SMTP.
- WLC 1.0 Backward Compatibility: Due to a number of changes, PIP implementations developed for WLC 1.0 are not supported under WLC 2.0. If you have created PIP implementations under WLC 1.0, you must reimplement them for WLC 2.0.

RosettaNet Administration

Administrative support for RosettaNet on BEA Weblogic Collaborate is via the WLC Administation console. For more information on adminstration tasks, see *Administering BEA WebLogic Collaborate*.

The WebLogic Collaborate Administration Console requires RosettaNet-specific configuration of document exchanges. For more information see the *BEA WebLogic Collaborate Administration Console Online Help*.

Configuring Collaboration Agreements for RosettaNet

For RosettaNet, a collaboration agreement is defined between two trading partners. It can be deployed with or without a hub. The following sections describe how to configure RosettaNet collaboration agreements for both configurations.

Peer-to-Peer with No Hub

In a hubless configuration, you are assumed to be running RosettaNet strictly on a peer-to-peer basis.

In such a situation, you would configure a collaboration agreement similar to the one shown in the following listing. In this case, the collaboration agreement is represented as XML data, though in practice you will configure the collaboration agreement using the WebLogic Collaborate Administration Console.

Listing 1-1 RosettaNet Collaboration Agreement Configuration with No Hub

```
<collaboration-agreement
  name="RN2|9.9|RosettaNet2|100"
  global-identifier="RN2|9.9|RosettaNet2|RNBuyer|RNSeller|102"
  version="1.0"
  status="ENABLED"
  conversation-definition-name="RN2"
  conversation-definition-version="2.0">
<party
   trading-partner-name="RNBuyer"
   party-identifier-name="RNBuyerPID"
   delivery-channel-name="RNBuyer"
   role-name="Buyer"/>
<party
   trading-partner-name="RNSeller"
   party-identifier-name="RNSellerPID"
   delivery-channel-name="RNSeller"
   role-name="Seller"/>
</collaboration-agreement>
```

- `global-identifier` is the system generated ID for the collaboration agreement.
- `conversation-definition-name` and the `role-name` are used to locate the PIP templates associated with the collaboration agreement.
- `party-identifer-name` is used to locate the business ID (DUNS number) of the trading partner.

Peer-to-Peer with Hub

You can also run RosettaNet on WebLogic Collaborate using a hub. In this configuration, the hub acts as an intermediary for the RosettaNet traffic. The hub makes no modification to the message being routed.

From the RosettaNet messages routed through the hub, the information accessible by the hub to locate collaboration agreement information is very minimal. Therefore, the collaboration agreement definitions on the hub are not used. They must still be defined, however, to allow the lower level transport routines to provide the client connections.

Instead, the hub uses one of four possible delivery channels for RosettaNet messages:

- RosettaNet-Hub-Channel—insecure channel for RNIF 1.1 messages.
- RosettaNet-Hub-SecureChannel—secure channel for RNIF 1.1 messages.
- RosettaNet2-Hub-Channel—insecure channel for RNIF 2.0 messages.
- RosettaNet2-Hub-SecureChannel—secure channel for RNIF 2.0 messages.

These are the channel names the hub will use. The channels must be created and configured using the WebLogic Collaborate Administration Console.

When a message arrives on an insecure channel, the hub will route the message to the appropriate trading partner's insecure delivery channel for the specific protocol version. If the trading partner contains more than one insecure channel, then the WebLogic Collaborate run-time will randomly pick one to use.

When a message arrives on a secure channel, the hub will route the message to the appropriate trading partner's secure delivery channel for the specific protocol version. If the trading partner contains more than one secure channel, then WebLogic Collaborate will randomly pick one to use.

For example, suppose a trading partner contains the following delivery channels defined on the hub.

Listing 1-2 Trading Partner Delivery Channel Definitions

```
<delivery-channel
  name="RNSellerSecureChannel"
  status="ENABLED"
  nonrepudiation-of-origin="true"
  nonrepudiation-of-receipt="true"
  secure-transport="true"
  confidentiality="true"
  transport-name="RNSellerTransport"
  document-exchange-name="DOCEX-RosettaNet2"/>
<delivery-channel
  name="RNSellerNonSecureChannel"
  status="ENABLED"
  nonrepudiation-of-origin="false"
  nonrepudiation-of-receipt="false"
  secure-transport="false"
  confidentiality="false"
  transport-name="RNSellerTransport"
  document-exchange-name="DOCEX-RosettaNet2"/>
```

When a message arrives at the hub for the trading partner, it would be routed in one of two ways:

- If the message arrived on a secure channel, then the message would be routed to the trading partner using `RNSellerSecureChannel`.
- If the message arrived on an insecure channel, the message would be routed to the trading partner using `RNSellerNonSecureChannel`.

In either case, any collaboration agreement information defined in the hub for the RosettaNet protocol would be ignored.

CHAPTER

2 Configuring RosettaNet Security

The following sections describe how to configure security under BEA WebLogic Collaborate for the RosettaNet business protocol:

- Configuring SSL and Digital Signatures
- Message Encryption

Configuring SSL and Digital Signatures

Both SSL and digital signature security for RosettaNet are configured through the WebLogic Collaborate Administration Console. For more information about configuring SSL and digital signatures, see the *BEA WebLogic Collaborate Administration Console Online Help*, *Using BEA WebLogic Collaborate Security*, and *BEA WebLogic Server Administration Guide*.

Message Encryption

The RosettaNet Implementation Framework 2.0 introduces a new security option to the overall RosettaNet framework. In contrast to RNIF 1.1, RNIF 2.0 allows encryption of a message at one of three levels:

- None - No encryption occurs.
- Payload - The service content and any attachments are encrypted.
- Entire Payload - The service header, service content, and attachments are all encrypted.

Encryption options can be configured through the WebLogic Collaborate Administration Console. For more information on configuration of the encryption options, see *Administering BEA WebLogic Collaborate*.

CHAPTER

3 Using Workflows with RosettaNet

The following sections explain how to develop workflows for WebLogic Collaborate for the RosettaNet business protocol:

- Understanding RosettaNet
- Understanding PIP Workflow Instances
- Getting Started
- Working with Workflows
- RosettaNet Template Variables
- Receiving a RosettaNet Message
- Sending a RosettaNet Message
- Validating a Message
- Message Attachments

These procedures refer to the workflow diagrams in the RosettaNet samples. To access a workflow diagram, run the WebLogic Process Integrator Studio. For information about defining WebLogic Collaborate workflows, see *Creating Workflows for BEA WebLogic Collaborate*.

Understanding RosettaNet

The following RosettaNet documents are required reading if you want to implement your own PIP using the support for RosettaNet provided by WebLogic Collaborate, and recommended reading if you want to fully understand the sample RosettaNet PIP implementations:

- *Creating Workflows for BEA WebLogic Collaborate*
- The following documents are available in the "Standards" section at the RosettaNet Web site (`http://www.rosettanet.org`):
  - RosettaNet Implementation Framework v1.1 (RNIF 1.1) —The RNIF is an open, common networked-application framework designed to allow RosettaNet Supply Chain and Solution Partners to collaborate in executing RosettaNet PIPs.
  - RosettaNet Implementation Framework v2.0 (RNIF 2.0)
  - RNIF Technical Advisories—RNIF Technical Advisories are updates and additional information for RNIF 1.1 and RNIF 2.0.
  - RNIF Technical Recommendations—Technical Recommendations describe features or enhancements not yet available in a published version of the RNIF v1.1. Implementation of Technical Recommendations is optional.
  - RNIF Business Signals, Service Header & Preamble—The RNIF Business Signals, Service Header & Preamble document contains message guidelines and XML document type definitions (DTDs) for the RNIF Business Signals, Service Header and Preamble.
  - Understanding a PIP Blueprint—reference for PIP blueprint components and evaluation. Available under Supporting Documents in the Standards Section.
  - PIPs of interest—PIPs are specialized system-to-system, XML-based dialogs that define business processes between supply chain companies. Each PIP includes a technical specification based on the RosettaNet Implementation Framework (RNIF), a Message Guideline document with a PIP-specific version of the Business Dictionary, and XML document type definitions (DTDs) for PIP-specific messages.

Understanding PIP Workflow Instances

WebLogic Collaborate implements the standard RosettaNet PIP definitions through the use of WebLogic Process Integrator public workflows. These workflow interface with other trading partners, while private workflows are used for message generation and resolution.

Figure 3-1 Message Workflow

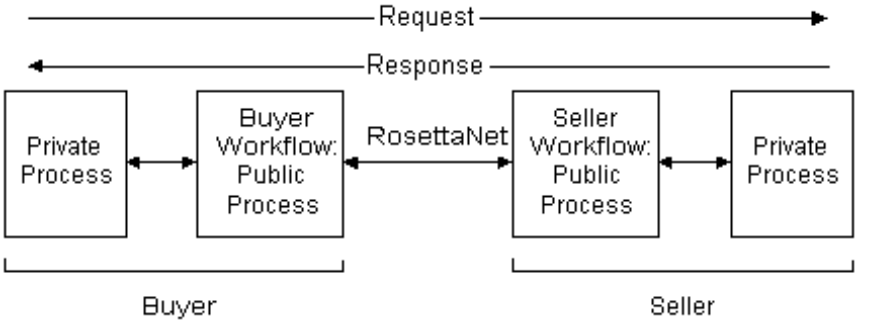


Figure 3-1 shows the process by which PIP workflows pass messages between trading partners. Generally speaking, RosettaNet-oriented workflows send messages as follows:

1. Buyer private process initiates a RosettaNet message. Data is retrieved and formatted into a RosettaNet message structure, and the appropriate PIP is determined.
2. Buyer public process creates RosettaNet message using appropriate PIP workflow. Message is sent to Seller.
3. Seller public process receives message, processes header information, and passes validated Buyer information, plus message content, to private process.
4. Seller private process resolves message content and generates a reply. The reply is processed into a RosettaNet message structure and passed to the public process.
5. Seller public process creates RosettaNet reply message and sends it to Buyer.

6. Buyer public process receives reply message, processes header information, and passes validated Seller information, plus message content, to private process.
7. Buyer private process resolves content of reply message.

Getting Started

Before you develop a PIP workflow, you should create a schema file from the PIP specification's DTD file. WebLogic Collaborate uses the schema file to validate RosettaNet messages. Schema files are not required if you do not intend to use validation. WebLogic Collaborate includes full implementations of PIPs 0A1 and 3A2, both of which can be used under both RosettaNet 1.1 and 2.0. You should use the schema files from these PIPs as a model for your own PIP schema file. These schema files are available in:

`WLC_HOME\rosettanet\schemas`

For more information on the use of custom schemas, refer to *Administering BEA WebLogic Collaborate*.

While most RosettaNet workflow development tasks are identical to those used for developing standard XOCP-oriented workflows, there are some differences. The standard procedures are documented in *Creating Workflows for BEA WebLogic Collaborate*; you should refer to them before creating RosettaNet workflows. The existing RosettaNet templates also provide a useful template base that you can use when creating new PIPs and workflows for them.

The remainder of this section discusses issues involving the creation of RosettaNet-specific workflows.

Working with Workflows

The RosettaNet PIP templates are public workflows, that allow you to connect to a trading partner. These workflows are designed to work with private workflows that:

- Provide raw source and recipient data
- Structure data into standard RosettaNet message body forms
- Process received data
- Initiate public workflows to send data
- provide an interface to private applications

Starting a Public Workflow

To start a public workflow from a private workflow within WebLogic Process Integrator, choose Actions → Collaborate → Start Public Workflow. For more information, see *Creating Workflows for BEA WebLogic Collaborate*.

Starting a Private Workflow

To start a private workflow from a public workflow within WebLogic Process Integrator, you must choose Actions → Start Workflow. For more information, see *Creating Workflows for BEA WebLogic Collaborate*.

RosettaNet Template Variables

RosettaNet workflows implemented under WebLogic Collaborate require a number of workflow variables to operate. These variables are used in three ways:

- Input variables are used to receive data into the workflow. The content of a variable is set by the calling private workflow.
- Output variables are used to send data out of the workflow. The content, created by the PIP when it completes, is passed back to the calling workflow.
- System variables are used to track within the workflow assorted data that are required for a RosettaNet workflow to operate properly.

All system variables are required for a RosettaNet workflow. Requirements for input variables are based on the actual input requirements of the PIP, but all RosettaNet workflows require the input variables listed in the following table. Output variables are also based on the requirements of individual PIPs, but the set shown in the table is required for all RosettaNet workflows.

Table 3-1 Required RosettaNet Workflow Variables

Workflow Variable	Type	Description	Usage	Supports
`attachmentDescriptorInput`	xml	The XML description of the attachments for a message	Input (optional)	RNIF 2.0
`fromContactName`	string	The sender's contact name	Input (mandatory)	RNIF 1.1
`fromDUNS`	string	The DUNS of the sender	Input (mandatory)	RNIF 1.1, RNIF 2.0
`fromEMail`	string	The sender's email	Input (mandatory)	RNIF 1.1
`fromLocation`	string	The location of the sender	Input (mandatory)	RNIF 2.0
`fromPhone`	string	The sender's phone number	Input (mandatory)	RNIF 1.1

Workflow Variable	Type	Description	Usage	Supports
fromSupplyChain	string	The supply chain code of the sender: *Information Technology* or *Electronic Components*	Input (mandatory)	RNIF 1.1
globalUsageCode	string	The global usage code: either *Test* or *Production*	Input (mandatory)	RNIF 1.1, RNIF 2.0
PIPInput	xml	The PIP message content to be sent; must match the PIP specification	Input (mandatory)	RNIF 1.1, RNIF 2.0
toDUNS	string	The DUNS of the receiver	Input (mandatory)	RNIF 1.1, RNIF 2.0
toLocation	string	The location of the receiver	Input (mandatory)	RNIF 2.0
toSupplyChain	string	The supply chain code of the receiver: *Information Technology* or *Electronic Components*	Input (mandatory)	RNIF 1.1
validateServiceContent	boolean	Flag to indicate whether the service content should be validated against the schema: True—validation required False—no validation required	Input (mandatory)	RNIF 1.1, RNIF 2.0
validateServiceHeader	boolean	Flag to indicate whether the service header should be validated against the schema: True—validation required False—no validation required	Input (mandatory)	RNIF 1.1, RNIF 2.0
NOF0A1Party1	string	The party name to be used for OA1 failure notification	Input (mandatory)	RNIF 1.1, RNIF 2.0
NOF0A1Party2	string	The party name to be used for OA1 failure notification	Input (mandatory)	RNIF 1.1, RNIF 2.0
attachmentDescriptorOutput	xml	The XML description of the attachments for a message	Output	RNIF 2.0

Workflow Variable	Type	Description	Usage	Supports
reason	string	Contains the reason for PIP exit: SUCCESS—if it was successful. Otherwise the reason description. If PIP 0A1 Notification of Failure is invoked, the failure reason will be returned in this field.	Output	RNIF 1.1, RNIF 2.0
PIPOutput	xml	The PIP message content received	Output	RNIF 1.1, RNIF 2.0
actionCode	string	The action code of a message	System	RNIF 1.1, RNIF 2.0
actionCodeVersion	string	The version of the action code	System	RNIF 1.1, RNIF 2.0
businessActivityID	string	The business activity of the message	System	RNIF 1.1, RNIF 2.0
docId	string	The document ID	System	RNIF 1.1
exceptionError	boolean	Indicates whether an exception signal was received	System	RNIF 1.1, RNIF 2.0
fromClass	string	The sender's partner classification code	System	RNIF 1.1
fromRole	string	The role of the sender	System	RNIF 1.1, RNIF 2.0
fromService	string	The sender's service code	System	RNIF 1.1, RNIF 2.0
functionCode	string	The global function code: *Request* or *Response*	System	RNIF 1.1
initiatingPartnerDUNS	string	The DUNS of the partner that initiated the process	System	RNIF 1.1, RNIF 2.0
inReplyToActionCode	string	In response to action code	System	RNIF 1.1, RNIF 2.0
inReplyToActionCodeVersion	string	In response to action code version	System	RNIF 1.1, RNIF 2.0

Workflow Variable	Type	Description	Usage	Supports
inReplyToMessageId	string	In response to message ID	System	RNIF 1.1, RNIF 2.0
isSignal	boolean	Indicates whether a signal was received	System	RNIF 1.1, RNIF 2.0
messageCode	string	The status of a message send	System	RNIF 1.1, RNIF 2.0
messageTrackingId	string	The ID of a message	System	RNIF 1.1, RNIF 2.0
PIP	string	The PIP name	System	RNIF 1.1, RNIF 2.0
PIPVersion	string	The version of the PIP	System	RNIF 1.1, RNIF 2.0
retryCount	integer	The retry count for the PIP	System	RNIF 1.1, RNIF 2.0
SERVICE_CONTENT_SCHEMA	string	The schemas used to validate the service content	System	RNIF 1.1, RNIF 2.0
serviceContent	xml	Temporary holder of the service content	System	RNIF 1.1, RNIF 2.0
signalCode	string	The code of the signal	System	RNIF 1.1, RNIF 2.0
signalCodeVersion	string	The version of the signal	System	RNIF 1.1, RNIF 2.0
timeout	boolean	Indicates whether a timeout has occurred	System	RNIF 1.1, RNIF 2.0
timeStamp	string	The timestamp for a message	System	RNIF 1.1, RNIF 2.0
toClass	string	The receiver's partner classification code	System	RNIF 1.1
toRole	string	The role of the receiver	System	RNIF 1.1, RNIF 2.0

Workflow Variable	Type	Description	Usage	Supports
`toService`	string	The service of the receiver	System	RNIF 1.1, RNIF 2.0
`transactionCode`	string	The global transaction code	System	RNIF 1.1
`validationError`	boolean	Indicates whether a validation error occurred	System	RNIF 1.1, RNIF 2.0

An important point to note when using these workflow variables is that all input variables must be initialized when you start a public process workflow. The values used to initialize the input variables do not need to be exposed or transmitted.

In addition, while you must set the NOF0A1 parties, there are no conventions for distinguishing among users in NOF0A1. Because the actual destination for the PIP is determined by the workflow, you may assign either identity to yourself or your trading partner.

Receiving a RosettaNet Message

WebLogic Collaborate supports two different methods of receiving RosettaNet messages—Start nodes and Event nodes. These two different node types are used depending on the circumstances under which the message is received.

Start Nodes

You can configure a workflow so that WebLogic Collaborate starts it automatically when WebLogic Collaborate receives the first message for a PIP instance. To configure this action, declare received PIPs as Start node events. Your receiving workflow starts and processes the incoming PIP. For an example, see the Participant Workflow discussion in Chapter 3, “Starting Collaborative Workflows,” of *Creating Workflows for BEA WebLogic Collaborate*.

The PIP3A2 Supplier workflow template serves as an example of how to configure a workflow template definition to be started by an incoming message. In this example, the starting event is set as a Collaboration Event. The workflow is automatically started and the output and system variables are set when a RosettaNet message has been received.

Event Nodes

Workflows can contain events that are triggered when a message is received for the PIP instance associated with the workflow. For an example, see the Initiator Workflow discussion in Chapter 3, "Starting Collaborative Workflows," of *Creating Workflows for BEA WebLogic Collaborate.*

Implementing Timeouts

Workflows can use an optional timeout path to wait for an incoming RosettaNet message. If the workflow waits for a response to a sent message (for example, the 3A2 Buyer workflow), you must create a separate timeout path to wait for the response. This path, illustrated in the 3A2 Buyer workflow template, consists of a timer, set for the appropriate timeout period, and a stop node.

Out of Sequence Reception of Signals

RNIF 1.1 and RNIF 2.0 define different standards for reception of signals and replies. These differing standards affect how your PIP workflow resolves these messages.

RNIF 1.1 specifies that replies must always follow signals. Thus, you might see a signal/response pattern as follows:

1. Initiator → (request) → recipient
2. Initiator ← (receipt acknowledgment) ← recipient
3. Initiator ← (response) ← recipient
4. Initiator → (handling acknowledgment) → recipient

RNIF 2.0 allows out-of-sequence receipt of signals. For example, a workflow might receive a reply before the receipt acknowledgment. Thus, RNIF 2.0 allows a signal/response pattern such as the following:

1. Initiator → (request) → recipient
2. Initiator ← (response) ← recipient
3. Initiator ← (receipt acknowledgment) ← recipient
4. Initiator → (receipt acknowledgment) → recipient

This pattern must be processed accurately in your workflow design. The PIP 3A2 Customer workflow template provides a useful example of how to handle this. The RNIF 1.1 version of the PIP template processes the signal, followed by the response. The RNIF 2.0 version provides logic to process the signal and the response separately.

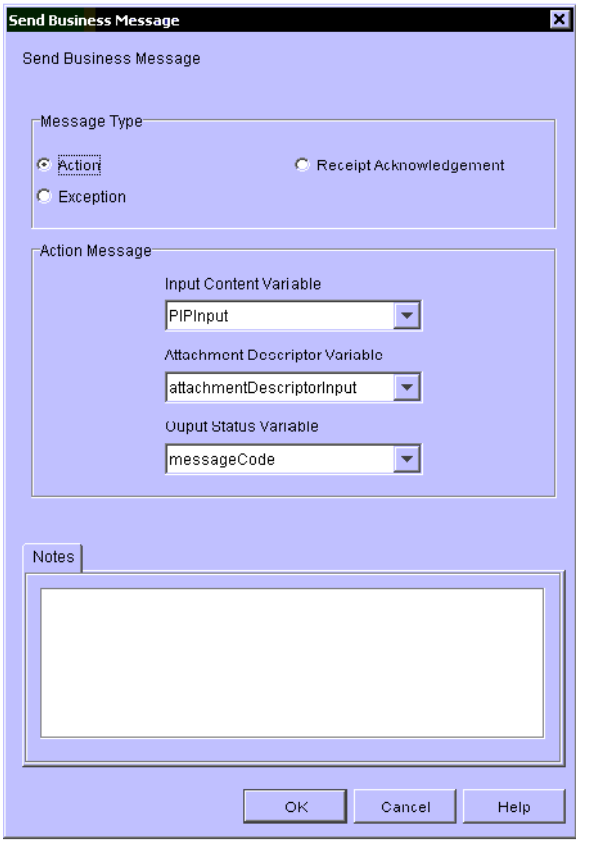
Sending a RosettaNet Message

A workflow can send a RosettaNet message by invoking an integration action called Send Business Message. To select this action:

1. Select Add Action from a node.
2. Choose Integration Actions→Collaborate→Send Business Message.

For example, when you choose the Send Business Message action, the following screen appears.

Figure 3-2 RosettaNet 2.0 Send Business Message Dialog



Depending on the message type that you select, various options are available for the message.

Message Type	Field Name	Description
Action	Message Type—Action	Indicates that a RosettaNet Action Business Message is to be sent.
Action	Input Content Value	A mandatory WebLogic Process Integrator workflow XML variable containing the XML data to represent the service content.
Action	Attachment Descriptor Value	An optional WebLogic Process Integrator workflow XML variable containing the XML data for describing the attachment(s) to be sent as part of the RosettaNet message.
Action	Output Status Variable	An optional WebLogic Process Integrator workflow Integer variable to contain the send status of the message.
Receipt Acknowledgment	Message Type—Receipt Acknowledgment	Indicates that a RosettaNet Receipt Acknowledgment signal is to be sent. No other data values are required.
Exception	Message Type—Exception	Indicates that a RosettaNet Receipt Exception Acknowledgment signal is to be sent.
Exception	Error Code	The Error Code (as defined by RosettaNet) to be sent.
Exception	Error Description	A brief description of the error.
Exception	Output Status Variable	An optional WebLogic Process Integrator variable to contain the send status of the message.

Validating a Message

The message validation process uses the Xerces 1.2.0 DOM parser, which supports an alpha implementation of the XML Schema specification. The Xerces 1.2.0 DOM parser is packaged with the WebLogic Collaborate software.

The following sections describe how WebLogic Collaborate validates RosettaNet messages and provide a bibliography for further reading about message validation:

- RosettaNet Message Validation
- Recommended Reading About Message Validation

RosettaNet Message Validation

BEA WebLogic Collaborate provides message validation services for both RNIF 1.1 and RNIF 2.0 messages. In either case, you use the appropriate DTDs to perform message validation. Depending on the values specified in the `validateServiceContent` and `validateServiceHeader` workflow variables, you may validate either, both, or neither:

- If `validateServiceContent` is true, all messages sent and received for a template are validated against the appropriate schema.
- If `validateServiceHeader` is true, the service header of all messages sent and received for a template are validated against the appropriate schema.

For an explanation of the exception handling process, see the RNIF at the following URL:

`http://www.rosettanet.org`

The example XML schema files and document type definition (DTD) files are located on your WebLogic Collaborate installation at:

`WLC_HOME``/rosettanet/schemas`

Recommended Reading About Message Validation

The following information is recommended reading to fully understand the example XML schemas; it is required reading if you are planning to implement your own XML schemas:

- Information about XML schema tools, usage, specifications, and development is available at the following URL:

  `http://www.w3.org/XML/Schema`

  XML Schema Part 0: Primer provides good descriptions of the features and capabilities of XML schema.

- Information about the Xerces implementation of the XML Schema specification is available at the following URL:

  `http://xml.apache.org/xerces-j/schema.html`

Performance Tuning and Message Validation

Message validation is used primarily while setting up and configuring your system with a partner. Once you are satisfied that invalid message generation issues have been removed from the system, you may optionally turn off message validation to boost performance. As noted elsewhere, there is no requirement for message validation to be performed during message processing. Rather, it is assumed that valid messages are being sent.

Message Attachments

RNIF 2.0 includes support for optional message attachments in the RosettaNet action message. Attachments are not file type-specific, and may contain binary data. Examples of possible attachments include Word documents, GIF images, PDF files, and so on. The information for each attachment is contained in the service header of the message.

BEA WebLogic Collaborate provides support for attachments by allowing a user application (for example a private workflow) to provide a description of the attachment, contained in a BEA-specific structured XML file, as an input to the PIP workflow. The XML file is a description of the file attachment. It is not the actual attachment. It only specifies which files should be attached by WebLogic Collaborate.

The following is the DTD information describing the attachments:

Listing 3-1 WebLogic Collaborate Attachment DTD Information for RosettaNet

```
<!ELEMENT WLCRosettaNet ( Attachment+ )>
<!ELEMENT Attachment (
                      description?,
                      Type,
                      Id,
                      LocalLocation
                      )>
<!ELEMENT description ( FreeFormText ) >
<!ELEMENT FreeFormText ( #PCDATA ) >
<!ATTLIST FreeFormText xml:lang CDATA #IMPLIED >
<!ELEMENT Type ( #PCDATA ) >
<!ELEMENT Id ( #PCDATA ) >
<!ELEMENT LocalLocation ( #PCDATA ) >
```

The following table describes the elements used in the DTD.

Table 3-2 RosettaNet Attachment Elements

Element	Description
Description	An optional description of the attachment.
Type	The MIME type qualifier code of the attachment.
ID	The Universal Resource Identifier for the attachment.

Table 3-2 RosettaNet Attachment Elements

Element	Description
LocalLocation	On sending: Contains the full path (on the local system) of the file for sending as an attachment. On receiving: Contains the full path (on the local system) of the file attachment that was received.

When the message is received, any attachments are stored locally in the *WLC_HOME*/runtime/rnattachments directory. The filename of the stored attachment is prefixed with a timestamp. For example, you might use the following XML description for an attachment.

Listing 3-2 Sample XML Attachment

```
<?xml version ="1.0"?>
<!DOCTYPE  WLCRosettaNet SYSTEM "WLCRosettaNet.dtd">
<WLCRosettaNet>
  <Attachment>
    <description>
      <FreeFormText>Product user guide in PDF</FreeFormText>
    </description>
    <Type>application/pdf</Type>
    <Id>"001801236324xyz@xyz.test.com"</Id>
    <LocalLocation>c:\pdf\myfile.pdf</LocalLocation>
  </Attachment>
  <Attachment>
    ...
  </Attachment>
</WLCRosettaNet>
```

This sample would generate the following data in the service header and in the MIME header.

Listing 3-3 Sample Service Output

```
<Attachment>
```

```
  <description>
    <FreeFormText>Product user guide in PDF</FreeFormText>
  </description>
  <GlobalMimeTypeQualifierCode>PDF
  </GlobalMimeTypeQualifierCode>
  <UniversalResourceIdentifier>
   cid:Attachment.001801236324xyz@xyz.test.com
  </UniversalResourceIdentifier>
</Attachment>
```

Listing 3-4 Sample MIME Header

```
Content-Type: application/pdf; name="myfile.pdf"
Content-ID: <Attachment.001801236324xyz@xyz.test.com>
Content-Description: Product user guide in PDF
```

4 RosettaNet PIP Templates

BEA WebLogic Collaborate includes two PIP templates with which you can immediately implement RosettaNet-based solutions. This section includes descriptions of both the contents of each PIP template and the information required to implement the PIP. Versions of both PIPs are provided for both RNIF 1.1 and RNIF 2.0.

PIP0A1: Notification of Failure

This PIP is a system-level RosettaNet PIP required for all implementations. It is used to transmit information in the event a process failure occurs. Normally, it does not require significant customization. Rather, data is passed to it from calling PIP workflows. For this reason, the appropriate workflow variables must be set in the calling workflow before the PIP0A1 workflow is called. Specifically, the variables listed in the following table must be set and imported to the PIP0A1_Admin template. All input variables must be initialized when the workflow is called. All other workflow variables should be referenced from Table 3-1.

Table 4-1 Required PIP Workflow Variables

Workflow Variable	Type	Description	Usage
`attachmentDescriptorInput`	xml	The XML description of the attachments for a message	Input (optional)
`fromContactName`	string	The sender's contact name	Input (mandatory)
`fromDUNS`	string	The DUNS of the sender	Input (mandatory)
`fromEMail`	string	The sender's email	Input (mandatory)
`fromLocation`	string	The location of the sender	Input (mandatory)
`fromPhone`	string	The sender's phone number	Input (mandatory)
`fromSupplyChain`	string	The supply chain code of the sender: *Information Technology* or *Electronic Components*	Input (mandatory)
`globalUsageCode`	string	The global usage code: either *Test* or *Production*	Input (mandatory)
`PIPInput`	xml	The PIP message content to be sent; must match the PIP specification	Input (mandatory)
`toDUNS`	string	The DUNS of the receiver	Input (mandatory)
`toLocation`	string	The location of the receiver	Input (mandatory)
`toSupplyChain`	string	The supply chain code of the receiver: *Information Technology* or *Electronic Components*	Input (mandatory)
`validateServiceContent`	boolean	Flag to indicate whether the service content should be validated against the schema: True—validation required False—no validation required	Input (mandatory)

Workflow Variable	Type	Description	Usage
validateServiceHeader	boolean	Flag to indicate whether the service header should be validated against the schema: True—validation required False—no validation required	Input (mandatory)
NOF0A1Party1	string	The party name to be used for OA1 failure notification	Input (mandatory)
NOF0A1Party2	string	The party name to be used for OA1 failure notification	Input (mandatory)

The NOF0A1Party1 and NOF0A1Party2 variables must be set, but you are not required to set them in any particular order.

PIP3A2: Query Price and Availability

PIP3A2 is provided in template form as an example of how to implement a PIP. There are two versions of this PIP (one implemented for RNIF 1.1, and one for RNIF 2.0), and each version is modeled using WebLogic Process Integrator for both the sender and receiver roles. Use the PIP template version appropriate for the version of RosettaNet that you are using.

When implementing PIP3A2, you must set the mandatory input variables, all of which are listed in the following table. These variables can be set from within the PIP workflow, or they can be set by a calling workflow. All input variables must be initialized when the workflow is called. All other workflow variables should be referenced from Table 3-1.

Table 4-2 PIP3A2 Required Input Workflow Variables

Workflow Variable	Type	Description	Usage
attachmentDescriptorInput	xml	The XML description of the attachments for a message	Input (optional)
fromContactName	string	The sender's contact name	Input (mandatory)
fromDUNS	string	The DUNS of the sender	Input (mandatory)
fromEMail	string	The sender's email	Input (mandatory)
fromLocation	string	The location of the sender	Input (mandatory)
fromPhone	string	The sender's phone number	Input (mandatory)
fromSupplyChain	string	The supply chain code of the sender: *Information Technology* or *Electronic Components*	Input (mandatory)
globalUsageCode	string	The global usage code: either *Test* or *Production*	Input (mandatory)
PIPInput	xml	The PIP message content to be sent; must match the PIP specification	Input (mandatory)
toDUNS	string	The DUNS of the receiver	Input (mandatory)
toLocation	string	The location of the receiver	Input (mandatory)
toSupplyChain	string	The supply chain code of the receiver: *Information Technology* or *Electronic Components*	Input (mandatory)
validateServiceContent	boolean	Flag to indicate whether the service content should be validated against the schema: True—validation required False—no validation required	Input (mandatory)

Workflow Variable	Type	Description	Usage
validateServiceHeader	boolean	Flag to indicate whether the service header should be validated against the schema: True—validation required False—no validation required	Input (mandatory)
NOF0A1Party1	string	The party name to be used for OA1 failure notification	Input (mandatory)
NOF0A1Party2	string	The party name to be used for OA1 failure notification	Input (mandatory)
attachmentDescriptorOutput	xml	The XML description of the attachments for a message.	Output
reason	string	Contains the reason for PIP exit: SUCCESS—if it was successful. Otherwise the reason description. If PIP 0A1 Notification of Failure is invoked, the failure reason will be returned in this field.	Output
PIPOutput	xml	The PIP message content received	Output

Modeling Other PIPs

The PIP3A2 templates are provided to demonstrate how a RosettaNet PIP is put together under WebLogic Collaborate. Because of the complexity of the PIP, BEA recommends that when creating a new PIP, you copy the PIP3A2 source files and modify the copies to create your new PIP.

You should note the following points when modeling other PIPs:

- Use input and output workflow variables to transport data between this PIP and other processes. Use input variables for the PIP to receive data from another workflow, and use output variables to send data to another workflow. You must

remember to initialize all mandatory input variables when you start a PIP workflow.

- Other workflows may initiate a PIP workflow. To configure this type of initiation, choose Actions→Collaborate→Start Public Workflow. Use this method when you are the trading partner initiating a PIP transaction. When you use this method, remember that you must manually enter all of the PIP workflow variables listed in Table 4-1.
- PIP workflows may initiate other workflows, such as private workflows. To configure this type of initiation, choose Actions→Start Workflow. Use this method when you are the trading partner receiving a PIP transaction, and you want to pass that data to a private workflow. Use the private workflow to connect, for example, to a back-end ERP system.

Index