



webMethods RosettaNet Module

User's Guide

VERSION 6.0.1

webMethods, Inc.
3930 Pender Drive
Fairfax, VA 22030
USA
703.460.2500
<http://www.webmethods.com>

webMethods Administrator, webMethods Broker, webMethods Developer, webMethods Installer, webMethods Integration Server, webMethods Mainframe, webMethods Manager, webMethods Modeler, webMethods Monitor, webMethods Trading Networks, webMethods Workflow, and the webMethods logo are trademarks of webMethods, Inc. "webMethods" is a registered trademark of webMethods, Inc.

All other marks are the property of their respective owners.

Copyright © 2003 by webMethods, Inc. All rights reserved, including the right of reproduction in whole or in part in any form.

webM-RN-UG-20030123

Contents

Chapter 1. About this Book	1
Welcome!	2
Conventions	2
Typographical Conventions	2
Program Code Conventions	4
Related Documentation	4
webMethods RosettaNet Module Documentation	4
Trading Networks Documentation	5
webMethods Integration Server and Developer Documentation	7
Modeler Documentation	10
Monitor Documentation	11
Viewing this Document	11
Printing this Guide	11
Chapter 2. Concepts	13
What is the webMethods RosettaNet Module?	14
webMethods RosettaNet Module Features	15
webMethods RosettaNet Module Packages	16
webMethods RosettaNet Module Design-Time Architecture/Components	17
webMethods RosettaNet Module Run-Time Architecture/Components	20
Process Overview	23
Chapter 3. Getting Started	25
Typical Procedure for Implementing a RosettaNet PIP	26
Step 1: Install the webMethods RosettaNet Module Packages	26
Step 2: Define Trading Partner Profiles	26
Step 3: Import the PIP Archives You Are Going To Use	26
Step 4: Define Internal TN XML Document Types	26
Step 5: Customize Your Trading Partner Agreements	27
Step 6: Write the Inbound and Outbound Mapping Services	27
Step 7: Write Error Handler Services	27
Step 8: Customize the Process Model Template You Are Going To Use	28
Step 9: Generate and Update Your Process Model	28
Step 10: Starting and Running a Conversation	28
Chapter 4. Defining Trading Partner Profiles in Trading Networks	29
What Is a Trading Partner?	30

Why Are Trading Partner Profiles Important?	30
Defining Your Enterprise Profile Using the Trading Networks Console	31
Required Profile Fields	31
Contact Information	31
Delivery Method Information	32
Security Information	32
Activating Your Enterprise Profile	32
Defining Your Trading Partners' Profiles	33
Required Profile Fields	33
Contact Information	33
Delivery Method Information	34
Security Information	34
Activating Your Trading Partners' Profiles	35
Chapter 5. Importing PIP Archives	37
What Is a PIP Archive?	38
Why Do I Need to Import a PIP Archive?	38
What Is In a PIP Archive?	38
PIPs and Process Models	39
Importing PIP Archives	39
Customizing PIP Archives	41
Customizing IS document types for Existing Standard RosettaNet PIPs	41
Implementing a Custom PIP	42
Exporting PIP Archives	42
Advantages and Disadvantages of Customizing and Exporting PIP Archives	45
Chapter 6. Defining Internal TN XML Document Types	47
What are TN XML Document Types?	48
TN XML Document Types Provided by WebMethods	48
Defining Your Own Internal TN XML Document Types	49
XML Queries Used to Extract Attributes From Business Documents	51
SenderID XML Query	51
ReceiverID XML Query	51
Chapter 7. Customizing Trading Partner Agreements	53
What is a Trading Partner Agreement?	54
How Does the webMethods RosettaNet Module Identify a TPA?	54
Modifying the Sender, Receiver, and Agreement ID	54
Modifying an Initiator's TPA	55
Modifying a Fulfiller's TPA	55
Field Descriptions	56

Distinguishing Between an Initiator's TPA and a Fulfiller's TPA	57
Parameter Settings	57
Chapter 8. Mapping a RosettaNet PIP Document	61
What Is "Mapping" a Business Document?	62
Why Do You Create an Outbound Mapping Service?	62
Why Do You Create an Inbound Mapping Service?	62
Example of Mapping a Business Document	63
Creating an Outbound Mapping Service	64
Input/Output to Use	64
Flow Operations to Use	65
Example of an Outbound Mapping Service: Initiator/Sender	66
Example of an Outbound Mapping Service: Fulfiller/Receiver	67
Creating an Inbound Mapping Service	67
Input/Output to Use	68
Flow Operations to Use	68
Example of an Inbound Mapping Service: Initiator/Sender	69
Example of an Inbound Mapping Service: Fulfiller/Receiver	70
Reusing Mapping Services	70
Chapter 9. Customizing a Process Model Template	73
What is a Process Model?	74
TN Roles and Focal Roles	75
Process Model Templates Provided by webMethods	76
Steps in the Process Model Templates That You Must Modify	77
Prerequisites for Customizing a Process Model Template	78
Customizing a Process Model Template	78
Assigning the Service that a Step Invokes	81
Setting Timeout and Retry Count Values	81
Chapter 10. Running and Monitoring a Conversation	83
Running a Conversation and the ConversationID	84
XML Queries Used To Extract and Generate the ConversationID	85
Running a Conversation Example	85
Starting a Conversation	86
Rejoining an Existing Conversation on the Fulfiller's Side	86
Rejoining an Existing Conversation on the Initiator's Side	87
Why Monitor a Conversation?	87
Sources of Conversation Status and Information	87
About Monitor	88
Archiving RNOs to the File System	89

- Chapter 11. Handling Errors and Exceptions** **91**
 - Handling Errors and Exceptions 92
 - What Is a Handler Service? 92
 - Handler Services Provided With the webMethods RosettaNet Module 92
 - When Might Errors or Exceptions Occur? 94
 - General Errors With RosettaNet PIP Documents 95
 - Errors When a Transaction Times Out: Notification of Failure 96
 - Errors When a Business Document Is Out of Sequence 97
 - Initiating an Error Process Manually 97
 - Where to Find Information About Errors 98

- Appendix A. Processing Large Business Documents** **99**
 - Limits to Processing Large Business Documents 100
 - The IData Objects Limitation 100
 - The Pipeline Limitation 100
 - Customizing to Enable Processing of Large Business Documents 101
 - Step 1: Separate the RosettaNet PIP Document into Non-Repeating and Repeating Parts ... 101
 - Step 2: Customize the Outbound Mapping Service 103
 - Step 3: Customize the Inbound Mapping Service 105
 - Step 4: Customize the Validation Service 106
 - Now That You Have Made Your Customizations, What Happens During Run Time? 108
 - Outbound Mapping of a Large Business Document 109
 - Inbound Validation of a Large Business Document 110
 - Large Business Document Sample 111
 - Running the PIP 3D8 Sample 111

- Index** **113**

About this Book

■ Welcome!	2
■ Typographical Conventions	2
■ Related Documentation	4
■ Viewing this Document	11
■ Printing this Guide	11

Welcome!


This guide describes how to implement and monitor RosettaNet Partner Interface Processes (PIPs) using the webMethods RosettaNet Module. To use this guide effectively, you should:




- Be familiar with the webMethods Integration Server, the Server Administrator, and webMethods Developer and understand the concepts and procedures described in the *webMethods Integration Server Administrator's Guide* and the *webMethods Developer User's Guide*.
- Be familiar with webMethods Trading Networks Console and understand the concepts and procedures described in the various *webMethods Trading Networks* guides.
- Be familiar with webMethods Modeler and understand the concepts and procedures described in the *webMethods Modeler User's Guide*.
- Have a basic knowledge of RosettaNet and RosettaNet terminology. For more information, see <http://www.rosettanet.org>.
- Have installed the webMethods Integration Server, the webMethods Developer, the webMethods Trading Networks (server side and console side) software, the webMethods Modeler (server side and client side) software, and the webMethods RosettaNet Module software.

Conventions

Typographical Conventions

This document uses the following typographical conventions:

Convention	Example
Procedures are designated by a blue box in the left column. Procedures are presented as a series of numbered steps.	 To create a flow service 1 On the File menu, click New .
Terms that identify elements, options, selections, and commands on the screen are shown in bold.	The Service field on the Properties tab specifies the name of the requested service.
Storage locations for services on the Integration Server are shown in a narrow font using the convention <i>folder.subfolder:service</i> .	<code>pub.client:smtp</code> sets a MIME-type e-mail message.

Convention	Example
Characters that you must type exactly are shown in a typewriter font.	Type: <code>*Administrators*</code>
Variable information that you must change (based on your specific situation or environment) is shown in italics.	Log on to the proxy server with: <code>USER <i>proxy_user</i></code> <code>PASS <i>proxy_password</i></code>
Input and output variables for a service are shown in italics.	A service in the flow takes a Record List called <i>LineItems</i> .
Messages that the system displays on the console are shown in a typewriter font.	The server returns the following error to the user: <code>Server has reached client limit.</code>
Keyboard keys are shown in uppercase.	Press ENTER; then press TAB.
Keys that you must press simultaneously are joined with the "+" symbol.	Press CTRL+ALT+M.
Directory paths are shown with the "\" directory delimiter unless the subject is UNIX specific. In these cases, the "/" is used. If you are working in a UNIX environment, substitute a "/" for the "\" shown in the procedures in this book.	<code>webMethods6\IntegrationServer\config</code>
Information that you must read before beginning a procedure or that alerts you to negative consequences of certain actions is presented using this notation.	 Important! If the folder is not already open in the Navigation Panel, open it before you start the following procedure.
Notes that provide related, but non-critical, information are presented using this notation.	 Note: When you start the product, you are prompted to log on to a webMethods Integration Server.
Helpful information (such as shortcuts and alternatives) is presented using this notation.	 Tip! You can also use CTRL+C to copy an object.

Program Code Conventions

For programming code and command syntax, this document uses the following typographical conventions:

Convention	Example
Keywords and values that you must type exactly as printed are shown in typewriter font.	<code>%CoSymbol%</code>
Variable values or parameters that you must supply are shown in italics.	<i>%VarName%</i>
Keywords or values that are optional are enclosed in []. Do not type the [] symbols in your own code.	<code>%loop LoopVar [null=NullValue]%</code>

Related Documentation

This section lists the documentation that webMethods provides with the webMethods RosettaNet Module. In addition, it lists documentation provided with webMethods Trading Networks, webMethods Integration Server, webMethods Developer, webMethods Modeler, and webMethods Monitor that you might find useful.

webMethods RosettaNet Module Documentation

Refer to this book...	For...
<i>webMethods RosettaNet Module Sample Guide</i>	Information about and instructions for running the PIP 3A4 sample that is included with the webMethods RosettaNet Module. Location: <code>ServerDirectory\packages\WmRosettaNet\publdoc\wMRN_SampleGuide.pdf</code>
<i>webMethods RosettaNet Module Built-In Services Reference Guide</i>	Information about the built-in services and records that are contained in the webMethods RosettaNet Module. Location: <code>ServerDirectory\packages\WmRosettaNet\publdoc\wMRN_BuiltInServices.pdf</code>

Trading Networks Documentation

The following table lists manuals that webMethods provides with Trading Networks. Some documents are in PDF format and others are in HTML.

Refer to this book...	For...
<i>webMethods Trading Networks—Getting Started with Trading Networks</i>	<p>An overview of webMethods RosettaNet Module, how to start and end webMethods RosettaNet Module, and information to familiarize you with the webMethods RosettaNet Module user interface.</p> <p>Location #1: webMethods6\TNConsole\doc\wMTN_GettingStarted.pdf</p> <p>Location #2: webMethods6\IntegrationServer\packages\WmTN\doc\wMTN_GettingStarted.pdf</p>
<i>Building Your Trading Network</i>	<p>Procedures for building your trading network, including how to define information about your trading partners, add partners to your trading network, and define the processing of business documents that are sent to your trading network.</p> <p>Location #1: webMethods6\TNConsole\doc\ wMTN_BuildingYourNetwork.pdf</p> <p>Location #2: webMethods6\IntegrationServer\packages\WmTN\doc\wMTN_BuildingYourNetwork.pdf</p>
<i>Managing and Analyzing Your Trading Networks</i>	<p>Procedures for updating information on your corporation and managing partners, and for analyzing the exchange of documents in your trading network, including searching and viewing documents that have flowed through your network and viewing audit logs of events that have occurred in your trading network system.</p> <p>Location #1: webMethods6\TNConsole\doc\ wMTN_AnalyzingYourNetwork.pdf</p> <p>Location #2: webMethods6\IntegrationServer\packages\WmTN\doc\wMTN_AnalyzingYourNetwork.pdf</p>

Refer to this book...	For...
<i>webMethods Trading Networks Programmer's Reference</i>	<p>Descriptions of the built-in service that you can use to programmatically access the functions of Trading Networks. This book is for developers.</p> <p>Location #1: webMethods6\TNConsole\doc\wMTN_ProgrammersRef.pdf</p> <p>Location #2: webMethods6\IntegrationServer\packages\WmTN\doc\wMTN_ProgrammersRef.pdf</p>
<i>webMethods Trading Networks API Reference</i>	<p>Descriptions of the Java classes that you can use to programmatically access the functions of Trading Networks. This reference is for developers who build services.</p> <p>Location: webMethods6\IntegrationServer\packages\WmTN\doc\api\index.html</p>
<i>webMethods Trading Networks Web Manager Configuration Guide</i>	<p>Description of Web Manager and instructions about how to configure Web Manager. This guide is for administrators of Trading Networks that want to learn the capabilities of Web Manager. It is also for developers that want to customize Web Manager.</p> <p>Location: webMethods6\IntegrationServer\packages\WmTNWeb\pub\doc\wMTN_WebMgrGuide.pdf</p>
<i>webMethods Trading Networks Large Document Handling</i>	<p>Description of how to set up Trading Networks so it is able to process large documents.</p> <p>Location #1: webMethods6\TNConsole\doc\wMTN_LargeDocHandling.pdf</p> <p>Location #2: webMethods6\IntegrationServer\packages\WmTN\doc\wMTN_LargeDocHandling.pdf</p>

webMethods Integration Server and Developer Documentation

The following table lists documentation that webMethods provides with webMethods Integration Server and Developer that you can use as references. Some documents are in PDF format and others are in HTML.

Refer to this book...	For...
<i>Introduction to Integration with webMethods</i>	An overview of the webMethods integration platform and how you use it to integrate business processes across applications, information systems, people, and companies. Location: webMethods6\Developer\doc\IntroToIntegration.pdf
<i>webMethods Integration Server Administrator's Guide</i>	Information about using the Server Administrator to configure, monitor, and control the webMethods Integration Server. This book is for server administrators. Location: webMethods6\IntegrationServer\doc\ISAdministratorsGuide.pdf
<i>webMethods Administrator User's Guide</i>	Information about using the centralized view provided by webMethods Administrator to access and administer remote servers securely on an enterprise network. This book is for server administrators. Location: webMethods6\IntegrationServer\packages\WmAdmin\pub\doc\ISWebmAdminUserGuide.pdf
<i>webMethods Developer User's Guide</i>	Information about using webMethods Developer to create and test services and client applications. This book is for developers. Location: webMethods6\Developer\doc\ISDeveloperGuide.pdf

Refer to this book...	For...
<i>webMethods Integration Server Clustering Guide</i>	Information about installing and using the webMethods Integration Server Clustering feature. It also contains information for administrators who configure and manage a webMethods Integration Server system and for application developers who want to create services that interact directly with the webMethods Integration Server Repository. Location: webMethods6\IntegrationServer\doc\ISClusteringGuide.pdf
<i>webMethods Built-In Services Reference Guide</i>	Descriptions of services that are installed on your Integration Server. This book for is for developers. Location: webMethods6\Developer\doc\ISBuiltInServicesGuide.pdf
<i>DSPs and Output Templates Developer's Guide</i>	Information about creating Dynamic Server Pages (DSPs) and output templates. This book is for developers. Location: webMethods6\Developer\doc\guides\ISDSPs&Templates.pdf
<i>webMethods Integration Server JAVA API Reference</i>	Descriptions of the Java classes you use to create services. This reference is for developers who build services using Java. Location: webMethods6\IntegrationServer\doc\api\Java\index.html
<i>webMethods Integration Server C/C++ API Reference</i>	Descriptions of the webMethods C/C++ application program interface. This reference is for developers who build services or clients with C or C++. Location: webMethods6\IntegrationServer\doc\api\C\index.html
<i>SOAP Developer's Guide</i>	Information about using the webMethods Integration Server to exchange SOAP messages. This book is for developers. Location: webMethods6\Developer\doc\guides\ISSoapGuide.pdf

Refer to this book...	For...
<i>Schema Reference Guide</i>	<p>Descriptions of the components of an IS schema and how the components relate to the XML Schema or Document Type Definition (DTD) from which it is generated. This book is for developers.</p> <p>Location:</p> <p>webMethods6\Developer\doc\guides\ISSchemaGuide.pdf</p>
<i>Flat File Schema Developer's Guide</i>	<p>Information about using flat file schemas and dictionaries to parse and validate inbound flat files and produce outbound flat files. This book is for developers.</p> <p>Location:</p> <p>webMethods6\Developer\plugins\FlatFilePlugin\doc\ISFlatFileSchemaGuide.pdf</p>
<i>MIME-S/MIME Developer's Guide</i>	<p>Information about using the webMethods Integration Server to construct MIME and S/MIME messages, secure them, transport them over the Internet, and extract information from them. This book is for developers.</p> <p>Location:</p> <p>webMethods6\Developer\doc\guides\ISMimeSmimeGuide.pdf</p>
<i>Guaranteed Delivery Developer's Guide</i>	<p>Information about using the guaranteed delivery features in the webMethods Integration Server to ensure guaranteed, one-time execution of services. This book is for developers.</p> <p>Location:</p> <p>webMethods6\Developer\doc\guides\GuaranteedDeliveryGuide.pdf</p>
<i>Web Services Developer's Guide</i>	<p>Information about using the webMethods Integration Server to create Web services and incorporate Web services into the solutions you develop. This book is for developers.</p> <p>Location:</p> <p>webMethods6\Developer\doc\guides\ISWebServicesGuide.pdf</p>
<i>Working with XML Documents</i>	<p>Information about using the webMethods Integration Server to send, receive, and query XML documents. This book is for developers.</p> <p>Location:</p> <p>webMethods6\Developer\doc\guides\ISXmlGuide.pdf</p>

Refer to this book...	For...
<i>webMethods Certificate Toolkit User's Guide</i>	<p>Information about installing and using the webMethods Certificate Toolkit. It also contains information for administrators and developers of webMethods components about creating and managing digital certificates for use with webMethods components.</p> <p>Location: webMethods6\CertToolkit\doc\ISCertToolkitGuide.pdf</p>
<i>webMethods Integration Server Administrator's Online Help</i>	<p>Information about the controls in the webMethods Server Administrator screens and step-by-step procedures describing how to perform tasks with the Server Administrator.</p> <p>You can access the online reference by clicking the Help link on a Server Administrator screen.</p>
<i>webMethods Developer Online Help</i>	<p>Information about the controls in the webMethods Developer application windows and step-by-step procedures describing how to perform tasks with the webMethods Developer.</p> <p>You can access the online reference by clicking Help in an application window or dialog box.</p>

Modeler Documentation

The following table lists the manuals that webMethods provides with Modeler.

Refer to this book...	For...
<i>webMethods Modeler User's Guide</i>	<p>Information about how to use Modeler to build, generate, and deploy business process models.</p> <p>Location: webMethods\Modeler\doc\ModelerUsersGuide.pdf</p>

Monitor Documentation

The following table lists the manuals that webMethods provides with webMethods Monitor.

Refer to this book...	For...
<i>webMethods Monitor User's Guide</i>	Information about monitoring business process models, services, and documents. Location: webMethods\ServerDirectory\packages\WmMonitor\MonitorUsersGuide.pdf

Viewing this Document

To view this document, which is in PDF format, you must have Acrobat Reader™ 4.0 or later installed on your system. If you have an earlier version of Acrobat Reader, you will receive the following error message when you open this document and Acrobat Reader will not display the images in this document:

Could not find the ColorSpace named 'Cs8.'

If you do not have this software or you do not have the correct version, you can download a free copy from:

<http://www.adobe.com/supportservice/custsupport/download.html>.

Printing this Guide

To produce a hard copy of this guide, print this document from Acrobat Reader.

Concepts

■ What is the webMethods RosettaNet Module?	14
■ webMethods RosettaNet Module Features	15
■ webMethods RosettaNet Module Packages	16
■ webMethods RosettaNet Module Design-Time Architecture/Components	17
■ webMethods RosettaNet Module Run-Time Architecture/Components	20
■ Process Overview	23

What is the webMethods RosettaNet Module?

The webMethods RosettaNet Module is an implementation of the RosettaNet Implementation Framework (RNIF), versions 1.1 and 2.0. Leveraging the webMethods platform process modeling and management capabilities and XML business document (in this guide called business document) recognition capabilities, the webMethods RosettaNet Module enables you to implement open and common e-business Partner Interface Processes (PIPs) and their implementation guidelines. The webMethods RosettaNet Module consists of a set of design-time and run-time components, both sets of which are discussed in this chapter. For information about design-time components, see [“webMethods RosettaNet Module Design-Time Architecture/Components”](#) on page 17. For information about run-time components, see [“webMethods RosettaNet Module Run-Time Architecture/Components”](#) on page 20.

You implement a PIP by importing the necessary PIP archive and then customizing one of four process model templates as described in [Chapter 9, “Customizing a Process Model Template”](#) in this guide. To customize a process model template, you assign services that you create to specific steps in the template. You also assign TN XML document types to steps that wait for business documents (in this guide called wait steps), so that when the webMethods RosettaNet Module receives a business document during a conversation, or process, it can recognize the type of business document received. For more information about process models and webMethods Modeler, the tool used to customize the process model templates, see the *webMethods Modeler User’s Guide*. For more information about PIP archives, see [Chapter 5, “Importing PIP Archives”](#) in this guide. For the typical procedure for implementing a RosettaNet PIP, see [Chapter 3, “Getting Started”](#) in this guide.

The process run time (PRT), a facility of the webMethods Integration Server, manages execution of RosettaNet conversations. The PRT is the run-time component that executes process logic, logs process data, and controls process execution order. During run time, the PRT uses the ConversationID associated with a business process to determine if the process is active, or currently running. For more information about the PRT, see [“webMethods RosettaNet Module Run-Time Architecture/Components”](#) on page 20 in this guide and the *webMethods Modeler User’s Guide*. For more information about ConversationIDs, see [“Running a Conversation and the ConversationID”](#) on page 84 in this guide.

Every PIP in the webMethods RosettaNet Module is associated with two Trading Partner Agreements (TPAs), one for the initiator/sender, in a conversation and one for the fulfiller/receiver, in a conversation. A TPA is a set of parameters that you can use to govern how business documents are exchanged between two trading partners. For example, a TPA for a RosettaNet conversation contains parameters that control such things as signing, encoding, and validation of RosettaNet PIP documents. For more information about TPAs, see [Chapter 7, “Customizing Trading Partner Agreements”](#) in this guide and the *Building Your Trading Network* manual.

webMethods RosettaNet Module Features

The webMethods RosettaNet Module, which runs on top of the webMethods Integration Server, contains these features:

- **Provides out-of-the-box support for the RNIF and RosettaNet PIPs**, which gives you the ability to quickly implement production solutions for automating the many interactions across your entire supply chain. The webMethods RosettaNet Module supports *Technical Advisories 1, 2, and 3*, as well as *Technical Recommendation 1, Use of File Attachments*, and *Technical Recommendation 2, Synchronous Responses*, put out by RosettaNet to enhance the RNIF 1.0 standard.
- **Enables you to create webMethods RosettaNet-based process models**. Use Modeler to create business process models that reflect your organization's business requirements.
- **Enables you to monitor your process models**. Use webMethods Monitor to manage and monitor your process models.
- **Allows you to incorporate new versions of PIPs** quickly after release from RosettaNet, without waiting for a new software release.
- **Enables you to implement multiple versions of PIPs**. You can implement multiple versions of the same PIP so that you can interact with multiple trading partners even though they may be using different versions of the same PIP. Note that you can communicate with a trading partner only if both of you are using the same version of the same PIP and RNIF. For example, you can communicate with Trading Partner A if both of you are using PIP 3A9 version 1.1, and you can communicate with Trading Partner B if both of you are using PIP 3A9 version 2.0, but you cannot communicate with Trading Partner A if Trading Partner A is using PIP 3A9 version 1.1 and you are using PIP3A9 version 2.0.
- **Allows you to capture trading partner-specific rules**, reflecting the unique business practices between your organization and your trading partners.
- **Allows you to leverage existing investments in enterprise solutions** by accepting information from EDI-based systems to populate business documents in RosettaNet format.
- **Supports the widest range of eStandards**, including cXML, CBL, OAG, FpML, BizTalk, Acord, EDI, and OBI, which gives you the ability to use a preferred approach in-house while supporting the different standards requirements of your customers. In addition, this allows you to use the webMethods RosettaNet Module as your platform for all eStandards interactions across the Internet.
- **Maintains transaction logging and audit trails to ensure the integrity of all trading partner transactions**. Automatic archival of transaction messages as well as digital signature support ensures non-repudiation of origin and content.
- **Incorporates custom PIPs**. Unlike other solutions, the webMethods RosettaNet Module does not require "hard coding" of PIPs. With the webMethods RosettaNet Module, you can implement custom PIPs if needed.

- **Supports synchronous and asynchronous transmission of PIP messages.** With the webMethods RosettaNet Module, you can employ either synchronous or asynchronous communications to better meet the time requirements of your trading partner transactions. For those PIPs that require immediate responses over the same Internet connections, you can use synchronous communications. For transactions with longer duration, you can use asynchronous communications.
- **Supports multi-byte languages,** such as Chinese and Japanese.
- **Supports large business documents.** The webMethods RosettaNet Module contains certain measures, such as the `pub.web:getNodeIterator` service, that enables the processing of large business documents.



Note: For information about the `pub.web:getNodeIterator` service, see the *webMethods Built-In Services Reference Guide*.

webMethods RosettaNet Module Packages

The webMethods RosettaNet Module contains several packages (sets of webMethods services and related files) that you install on the webMethods Integration Server. The following table describes the contents of each package. For detailed information about the contents of a package, see the *webMethods RosettaNet Module Built-In Services Reference Guide*.

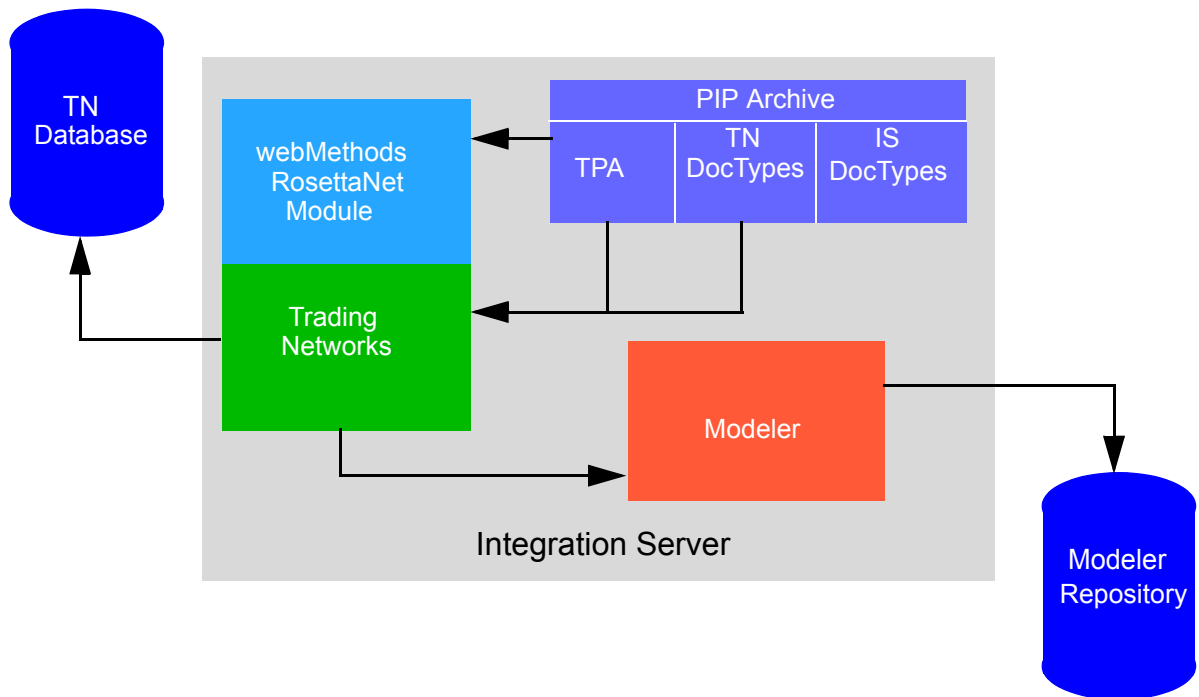
Package	Description
WmIPRoot	Contains the basic records and services that are commonly used by all of the other webMethods RosettaNet Module packages.
WmRosettaNet	Contains import/export functionality for PIPs and the <code>wm.ip.m:receive</code> service.
WmRNIF11TRP	Contains the records and services that implement the RNIF version 1.1 transport protocol.
WmRNIF20TRP	Contains the records and services that implement the RNIF version 2.0 transport protocol.
WmRNPips	Contains the records for the specific PIPs that you have imported into your webMethods RosettaNet Module installation. Note: This package is automatically created when you import the first PIP.

Package	Description
WmRNSample	Contains the records and services that implement the PIP 3A4 and synchronous PIP 2A9 samples, which are documented in the <i>webMethods RosettaNet Module Sample Guide</i> .
WmRNLargeDocSample	Contains the records and services that demonstrate the handling of large business documents.

webMethods RosettaNet Module Design-Time Architecture/Components

The following figure illustrates the webMethods RosettaNet Module design-time architecture and components, and the component relationships. For further explanation, see the table that follows the figure.

webMethods RosettaNet Module Design-Time Architecture/Components



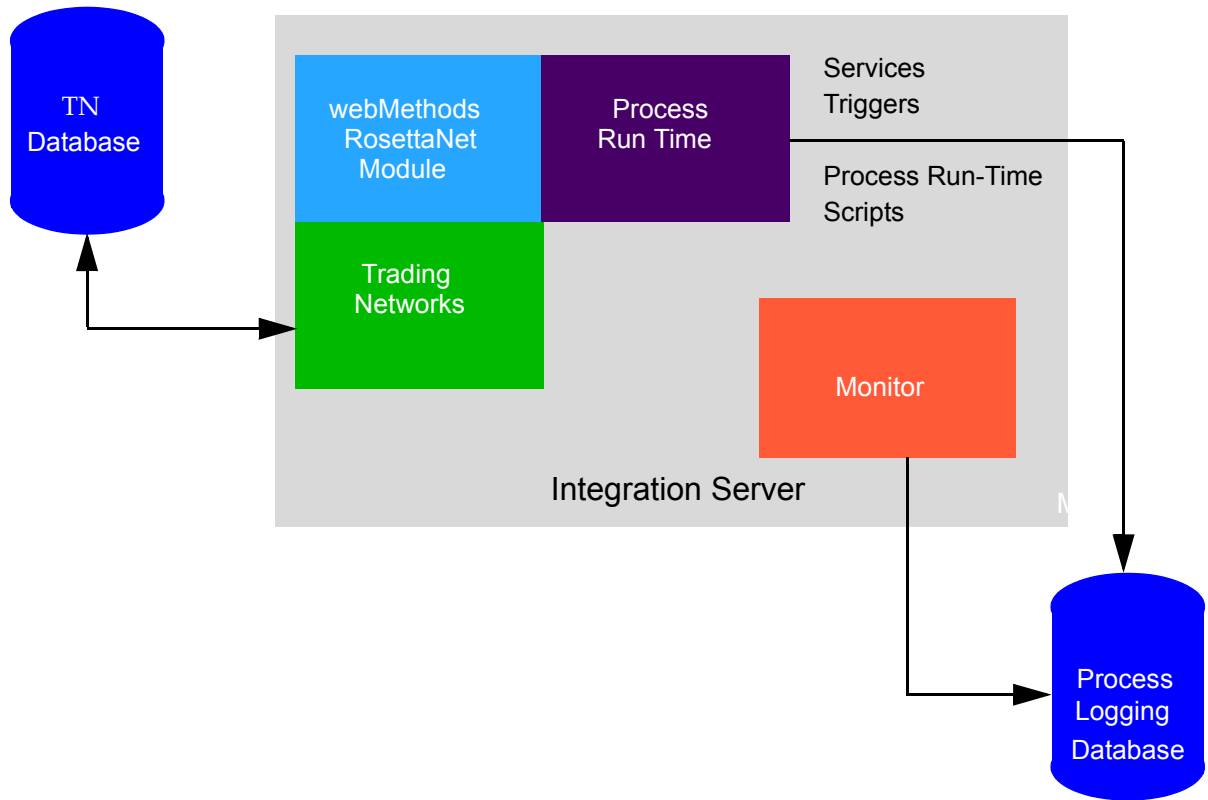
Component	Description
PIP Archive	<p>A Partner Interface Process (PIP) archive, also called a process archive, is a compressed file (.par) that contains two Trading Partner Agreements (TPAs) (one for the initiator/sender and one for the fulfiller/receiver of a conversation), TN XML document types, and IS document types for a specific version of a particular PIP. During design time, you import PIP archives for the PIPs you need into the Integration Server.</p>
webMethods RosettaNet Module	<p>The webMethods RosettaNet Module contains process model templates for the different kinds of PIPs you can implement. You can use these process model templates to define your PIPs from within Modeler.</p>
webMethods Trading Networks	<p>Trading Networks enables your enterprise to link with trading partners with whom you want to exchange business documents, thereby forming a business-to-business trading network. During design time, you define your trading partner profiles in the Trading Networks Console. The profiles contain the information that Trading Networks needs to exchange business documents with your trading partners.</p> <p>In addition to defining trading partner profiles during design time, you also customize the TPAs in the Trading Networks Console. And you can view TN XML document types as well. The TPAs and TN XML document types (both internal and external) are created in Trading Networks when you import the PIP archive files into the Integration Server.</p> <p>For more information about Trading Networks, trading partner profiles, TN XML document types, and TPAs, see the <i>Building Your Trading Network</i> manual. You can also find information about trading partner profiles in Chapter 4, “Defining Trading Partner Profiles in Trading Networks” in this guide and information about TPAs in Chapter 7, “Customizing Trading Partner Agreements” in this guide.</p>
Trading Networks Database	<p>Trading Networks saves trading partner profile, TN XML document type, and TPA information, among other things, to its database and retrieves this information when needed.</p>

Component	Description
Modeler	Modeler is a Java GUI. You use Modeler to customize the process model templates provided by webMethods, thereby creating your own process models. You customize a process model template by specifying how the process model is to interact with your back-end systems and editing the services that are invoked by the steps of the process model, among other things. When you generate a process model, you generate the run-time elements (services, triggers, and process run-time scripts, or fragments) that execute at run time. For more information about Modeler, see the <i>webMethods Modeler User's Guide</i> . For information about customizing a process model template, see Chapter 9, "Customizing a Process Model Template" in this guide.
Modeler Repository	The Modeler Repository is a storage area that Modeler uses to save process model information. For more information about Modeler, see the <i>webMethods Modeler User's Guide</i> .
Integration Server	The Integration Server contains the documents, services, and records that you will want to access when creating your process models.

webMethods RosettaNet Module Run-Time Architecture/Components

The following figure illustrates the webMethods RosettaNet Module run-time architecture and components, and component relationships. For further explanation, see the table that follows the figure.

webMethods RosettaNet Module Run-Time Architecture/Components



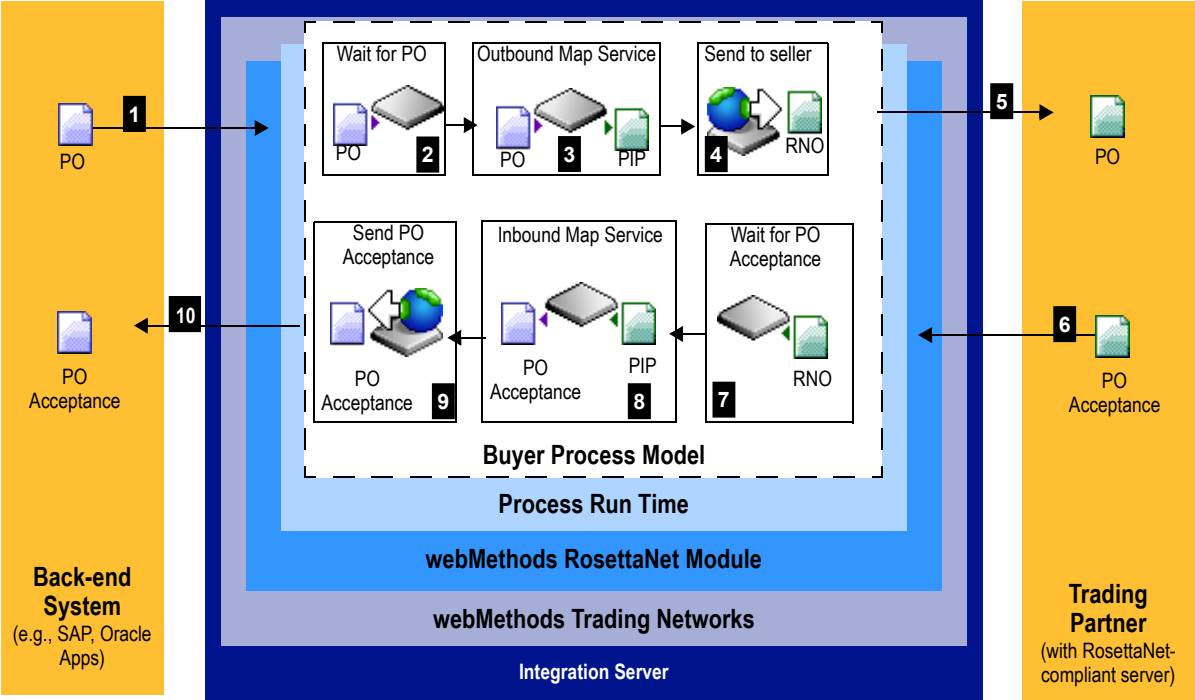
Component	Description
webMethods RosettaNet Module	<p>During run time, the webMethods RosettaNet Module receives a business document from a back-end system or trading partner. It invokes a Trading Networks service to recognize the business document, create a BizDocEnvelope and ConversationID, and save the BizDocEnvelope to the Trading Networks database. The webMethods RosettaNet Module passes the business document to the process run time (PRT). For more information about the PRT, see the PRT row in this table and the <i>webMethods Modeler User's Guide</i>.</p>
Trading Networks	<p>Trading Networks enables your enterprise to link with trading partners with whom you want to exchange business documents, thereby forming a business-to-business trading network.</p> <p>During run time, the webMethods RosettaNet Module uses Trading Networks services and TN XML document types to recognize business documents it receives, create BizDocEnvelopes, and save BizDocEnvelopes to the Trading Networks database. The webMethods RosettaNet Module uses the trading partner profiles in Trading Networks to know, for example, the methods by which to send business documents to its trading partners. The webMethods RosettaNet Module uses TPAs in Trading Networks to know information such as whether an outbound business document should be signed, whether the Service Header of the business document should be encrypted, along with any attachments, if present, and so on.</p> <p>For more information about Trading Networks, trading partner profiles, TN XML document types, and TPAs, see the <i>Building Your Trading Network</i> manual. You can also find information about trading partner profiles in Chapter 4, "Defining Trading Partner Profiles in Trading Networks" in this guide and information about TPAs in Chapter 7, "Customizing Trading Partner Agreements" in this guide.</p>
Trading Networks Database	<p>The Trading Networks database stores TN XML document type, TPA, and trading partner profile information, among other things.</p>

Component	Description
PRT	<p>The process run time (PRT) is a facility of the Integration Server that manages the execution of RosettaNet conversations. The PRT ensures the integrity, traceability, observability, and controllability of RosettaNet conversations by performing the following functions:</p> <ul style="list-style-type: none"> ■ Accepts business documents from Trading Networks. ■ Determines what process model to use for a given PIP transaction. ■ Processes a business document, based on the type of business document received and who sent it. ■ Records the status of the business document to the Process Logging Database. <p>During processing of a business document, the PRT uses the ConversationID associated with the business document to see if the business document belongs to a new or existing conversation. The PRT looks for a matching ConversationID among the enabled process models. If it does not find a matching ConversationID, it starts a new conversation (process). If the PRT finds an enabled process model with a matching ConversationID, the business document rejoins the conversation.</p> <p>For more information about the PRT, see the <i>webMethods Modeler User's Guide</i>. For more information about ConversationIDs, see "Running a Conversation and the ConversationID" on page 84 in this guide.</p>
webMethods Monitor	<p>You use Monitor to manage and monitor conversations. The Monitor displays information about a conversation by retrieving information from the Process Logging Database.</p> <p>You can manage a conversation by performing such commands as suspend, resume, restart, and terminate.</p>
Process Logging Database	<p>The Monitor and PRT log audit data about running conversations to the Process Logging Database.</p>
Integration Server	<p>The Integration Server contains the run-time elements that were generated from the automated controlled steps within the process model. The run-time elements are services, triggers, and process run-time scripts, or fragments.</p>

Process Overview

The following figure illustrates the use of PIP 3A4 *Manage Purchase Order* to send a business document to a trading partner using the webMethods RosettaNet Module. Only the conversation for the initiator/buyer role is displayed. For further explanation, see the table that follows the figure.

Initiator/Buyer Process Overview of PIP 3A4



Step	Description
1	Your enterprise (acting as an initiator/buyer) submits an internal PO from your back-end system to the webMethods RosettaNet Module. Upon receiving the internal PO, the webMethods RosettaNet Module invokes a Trading Networks service to recognize the TN XML document type. Trading Networks wraps the PO in a BizDocEnvelope and saves the BizDocEnvelope to the Trading Networks database. Using the BizDocEnvelope, the webMethods RosettaNet Module generates a ConversationID.
2	The webMethods RosettaNet Module passes the BizDocEnvelope to the PRT. The PRT looks for a matching ConversationID among the enabled process models. If the PRT does not find a matching ConversationID, it starts the conversation (process). At the Wait for PO step, a service retrieves the Trading Networks profiles for the sender and receiver and the TPAs.
3	An outbound mapping service maps the internal PO into a RosettaNet PIP.
4	At the Send to Seller step, a service packages the PIP into a RosettaNet object (RNO) and sends the RNO to the trading partner (seller).
5	The trading partner receives the PO, validates it, and sends a Receipt Acknowledgment (not illustrated).
6	The trading partner (seller) sends a PO Acceptance in the form of an RNO to your enterprise by way of the webMethods RosettaNet Module. The webMethods RosettaNet Module receives the PO Acceptance and extracts the ConversationID. The webMethods RosettaNet Module passes the PO Acceptance to the PRT.
7	The PRT matches the ConversationID from the PO Acceptance with the ConversationID associated with a specific process model. The conversation resumes, validation of the PO Acceptance occurs, and your enterprise sends an Acceptance Acknowledgment (not illustrated) to the trading partner.
8	An inbound mapping service maps the PO Acceptance from a PIP to an internal format recognizable by your back-end system.
9	At the Send PO Acceptance step, a service sends the PO Acceptance to your back-end system.
10	Your back-end system receives the PO Acceptance.

Getting Started

- Typical Procedure for Implementing a RosettaNet PIP 26

Typical Procedure for Implementing a RosettaNet PIP

The following procedure outlines the steps you take to implement a RosettaNet PIP.



Important! The following procedure assumes that you already have installed the webMethods Integration Server, webMethods Trading Networks, webMethods Modeler, and the webMethods RosettaNet Module.

Step 1: Install the webMethods RosettaNet Module Packages

You install the packages using the webMethods installer. The packages you install are:

- WmIPRoot
- WmRNIF11TRP
- WmRNIF20TRP
- WmRosettaNet
- WmRNSample (optional)
- WmRNLargeDocSample (optional)

Step 2: Define Trading Partner Profiles

In the webMethods Trading Networks Console, you define the profiles for yourself and all trading partners with whom you want to exchange business documents. A profile includes parameters such as the D-U-N-S[®] Number and the server URL. For more information about defining trading partner profiles, see [Chapter 4, “Defining Trading Partner Profiles in Trading Networks”](#) in this guide and the *Building Your Trading Network* manual.

Step 3: Import the PIP Archives You Are Going To Use

You download the compressed PIP archives, also called process archives, that represent the various versions of the PIPs that you want to implement. You install these archives on the Integration Server using the import functionality in the webMethods RosettaNet Module. For more information about importing PIP archives, see [Chapter 5, “Importing PIP Archives”](#) in this guide.

Step 4: Define Internal TN XML Document Types

Both internal and external TN XML document types are included in the PIP archives that you import to the Integration Server. When you import a PIP archive, the TN XML

document types for the respective PIP are automatically created in Trading Networks. However, there might be times that you will want to create your own internal TN XML document types or modify the provided internal TN XML document types. For more information about defining your own TN XML document types, see [Chapter 6, “Defining Internal TN XML Document Types”](#) in this guide.



Important! You must use the provided external TN XML document types, and you must not modify them. If you modify a provided external TN XML document type, the incoming business document will not join the conversation.

Step 5: Customize Your Trading Partner Agreements

You customize your trading partner agreements (TPAs) in the webMethods Trading Networks Console. A TPA is a set of parameters that you can use to govern how documents are exchanged between two trading partners. You customize a TPA by specifying the Sender and Receiver, and for an initiator’s TPA, by also specifying the Agreement ID. You also customize a TPA by modifying certain parameter values, such as whether you want an outbound business document to be encrypted or signed. For more information about TPAs, see [Chapter 7, “Customizing Trading Partner Agreements”](#) in this guide and the *Building Your Trading Network* manual.

Step 6: Write the Inbound and Outbound Mapping Services

You create mapping services, which consist of two types: outbound and inbound. You create an outbound mapping service to map an internal business document received from the back-end to a RosettaNet PIP document that you want to send to a trading partner. You create an inbound mapping service to map a RosettaNet PIP document received from a trading partner to an internal business document that is then sent to a back-end system. For more information about mapping business documents, see [Chapter 8, “Mapping a RosettaNet PIP Document”](#) in this guide.

Step 7: Write Error Handler Services

You write error handler services, if necessary. The webMethods RosettaNet Module provides default handler services for various tasks, such as handling a RosettaNet PIP document that does not conform to any TN XML document type. You can customize some of these handler services or create your own. For more information about error handling services, see [Chapter 11, “Handling Errors and Exceptions”](#) in this guide.

Step 8: Customize the Process Model Template You Are Going To Use

You determine the webMethods-provided process model template you need to use and then customize the template based on the PIP you are implementing. Customizing a process model template consists of such tasks as specifying the TN XML document type for a particular wait step, assigning inbound and outbound mapping services to the respective mapping steps, and so on. For more information about customizing process model templates, see [Chapter 9, “Customizing a Process Model Template”](#) in this guide. For more information about process models in general, see the *webMethods Modeler User’s Guide*.

Step 9: Generate and Update Your Process Model

You generate and update your process model in Modeler. For information about generating and updating your process model, see the *webMethods Modeler User’s Guide*.

Step 10: Starting and Running a Conversation

To start a conversation on the initiator’s side, the initiator’s back-end system calls `wm.ip.cm:process Document`. To start a conversation on the fulfiller’s side, a service on the initiator’s side sends a business document (as a RosettaNet object (RNO)) to `wm.ip.n:receive`. After you start a conversation, or process, you can monitor its progress using webMethods Monitor.

For more information about `wm.ip.cm:process Document` and `wm.ip.n:receive`, see the *webMethods RosettaNet Module Built-In Services Reference Guide*. For more information about starting a conversation, see [“Starting a Conversation” on page 86](#) in this guide. For more information about Monitor, see [“About Monitor” on page 88](#) in this guide, and the *webMethods Monitor User’s Guide*.

Defining Trading Partner Profiles in Trading Networks

- What Is a Trading Partner? 30
- Why Are Trading Partner Profiles Important? 30
- Defining Your Enterprise Profile Using the Trading Networks Console 31
- Defining Your Trading Partners' Profiles 33

What Is a Trading Partner?

A trading partner is any person or organization with whom you want to conduct business electronically. In the webMethods RosettaNet Module, a trading partner is defined by several criteria that you specify in a trading partner profile, including company name and identifying information, contact information, and preferred delivery methods.

In addition to specifying trading partner profiles for all of your trading partners, you must specify a profile for your own organization.

Why Are Trading Partner Profiles Important?

Your trading partner profiles, used in conjunction with trading partner agreements (TPAs) and process models, define how you and your trading partners exchange business documents. The different roles, for example, initiator/buyer or fulfiller/seller, in a process model define what actions your company can take in certain transactions, as well as the actions you expect your trading partners to perform during those transactions. In fact, the concise definition of profiles, the configuration of process models, the application of TPAs, and the implementation of the PIPs are what enables you to successfully interact with your trading partners.

You are likely to need to act in different roles with different trading partners. For example, you might need to act as a buyer toward your suppliers and as a seller toward your distributors. With the webMethods RosettaNet Module, you define a single trading partner profile for yourself (**My Enterprise**) and then customize the pertinent process model template to define each role you will perform in conducting transactions with your trading partners. You also must define a trading partner profile for each trading partner with whom you will conduct transactions, and again customize the pertinent process model template for each role those partners might play.

A trading partner's role figures prominently when you customize the process model templates for PIP transactions. When you customize a process model template, you will use the various trading partner profiles to help define the sender of the business document, the receiver of the business document, the target server's URL, and other criteria that depend heavily on the role of the trading partner. For information about customizing process model templates, see [Chapter 9, "Customizing a Process Model Template"](#) in this guide. For information about process models in general, see the *webMethods Modeler User's Guide*.

Defining Your Enterprise Profile Using the Trading Networks Console

Before defining your trading partner profiles in Trading Networks and exchanging business documents with your trading partners, you must first define your enterprise (**My Enterprise**) profile. You define your enterprise profile by completing the fields in the Profile Assistant in the Trading Networks Console. For procedural information about defining your enterprise profile as well as descriptions of the fields you must complete when defining your enterprise profile, see the *Building Your Trading Network* manual. For an example of completing your enterprise profile, see the *webMethods RosettaNet Module Sample Guide*.

The following sections specify the required fields you must complete to define your enterprise profile and RosettaNet-specific notes on other information you must complete, such as items you should know when completing contact information.

Required Profile Fields

Profile information is displayed on the **Corporate** tab. The following table lists and describes the required fields you must complete when defining your enterprise profile.

Required Profile Field for My Enterprise	Description
Corporation Name	The name of your enterprise.
External ID Type Value	Your enterprise's D-U-N-S® Number.

For descriptions of other fields you complete when you define your enterprise profile, see the *Building Your Trading Network* manual.

Contact Information

Contact information is displayed on the **Contact** tab.

If you are going to use RNIF version 2.0 as your transport protocol, you should enter at least one set of contact information. Be sure to include a valid e-mail address and phone number. The webMethods RosettaNet Module uses this information when using RNIF version 2.0 for processing a PIP 0A1 *Notification of Failure* when you have problems communicating with your trading partners.

Delivery Method Information

Delivery method information is displayed on the **Delivery Method** tab. To complete information on the **Delivery Method** tab, you need to know the following:

- You must specify at least one delivery method as your preferred method by selecting the **Use as preferred protocol** check box. If you do not specify a preferred delivery method in Trading Networks, the webMethods RosettaNet Module will not be able to determine how to send and receive RosettaNet PIP documents and therefore will not be able to execute any conversations.
- If you select one of the HTTP protocols as the delivery method, you must specify `invoke/wm.ip.rm/receive` as the location.
- If you would like to use an SMTP server other than that specified for error-specific e-mail messages in the Integration Server configuration, you must edit the `transport.cnf` file in the `WmRNIF20TRP` package and enter the name of the SMTP server you want to use in the `Smtphost` record. If you do not specify an SMTP server in the configuration file, RosettaNet uses the one specified in the Integration Server configuration information.



Note: You can find the `transport.cnf` file at:

```
webMethods6\IntegrationServer\packages\WmRNIF20TRP\config
```


where `webMethods6\IntegrationServer` is the directory in which the Integration Server is installed.

- You must also ensure that an e-mail account and listening service are configured to route incoming e-mails to your process. You can set this information up in the Server Administrator under **Security** ▶ **Ports** ▶ **Add Port** ▶ `webMethods/Email`. For detailed instructions, see the *webMethods Integration Server Administrator's Guide*.

Security Information

The webMethods RosettaNet Module uses the same certificate information for signing and decrypting.

Activating Your Enterprise Profile

After you define your enterprise profile, you activate it by clicking **Enable**  .

Defining Your Trading Partners' Profiles

Each trading partner with whom you want to exchange business documents must have a trading partner profile in Trading Networks. After you have defined your enterprise profile, you are ready to define your trading partners' profiles.

You define a trading partner profile by completing the fields in the Profile Assistant in the Trading Networks Console. For procedural information about defining a trading partner profile as well as descriptions of the fields you must complete when defining a trading partner profile, see the *Building Your Trading Network* manual.

The following sections specify the required fields you must complete to define a trading partner profile and RosettaNet-specific notes on other information you must complete, such as items you should know when completing security information.

Required Profile Fields

Profile information is displayed on the Corporate tab. The following table lists and describes the required fields you must complete when defining a trading partner profile.

Required Profile Field for Trading Partner	Description
Corporation Name	The name of the trading partner.
Partner Type	The type of trading partner. You should select webMethods Trading Networks .
External ID Type Value	Your trading partner's D-U-N-S Number.

For descriptions of other fields you complete when you define a trading partner profile, see the *Building Your Trading Network* manual.

Contact Information

Contact information is displayed on the **Contact** tab.

If you are going to use RNIF version 2.0 as your transport protocol with a trading partner, you should enter at least one set of contact information. Be sure to include a valid e-mail address and phone number. The webMethods RosettaNet Module uses this information when using RNIF version 2.0 for processing a PIP 0A1 *Notification of Failure* when you have problems communicating with your trading partners.

Delivery Method Information

Delivery method information is displayed on the **Delivery Method** tab. To complete information on the **Delivery Method** tab, you need to know the following:

- You must specify at least one delivery method as your preferred method by selecting the **Use as preferred protocol** check box. If you do not specify a preferred delivery method, the webMethods RosettaNet Module will not be able to determine how to send and receive business documents and therefore will not be able to execute any conversations.
- If HTTP is the preferred protocol, then specify the URL provided by your trading partner to receive RosettaNet PIP documents sent by you.
- If SMTP is the preferred protocol, then enter the e-mail address provided by your trading partner to receive RosettaNet messages sent by you. Additionally, if you would like to use an SMTP server other than that specified for error-specific e-mail messages in the Integration Server configuration, you must edit the `transport.cnf` file in the `WmRNIF20TRP` package and enter the name of the SMTP server you want to use in the `Smtphost` record. If you do not specify an SMTP server in the configuration file, RosettaNet uses the one specified in the IS configuration information.



Note: You can find the `transport.cnf` file at:

`webMethods6\IntegrationServer\packages\WmRNIF20TRP\config`

where `webMethods6\IntegrationServer` is the directory in which the Integration Server is installed.

Security Information

The Profile Assistant for defining a trading partner profile contains four security information screens. In these screens, you specify the security information you want to use when sending business documents to your trading partner.


The certificate names you enter in the fields on these screens are used for signing RosettaNet PIP documents you send to your trading partner and for decrypting encrypted RosettaNet PIP documents you receive from your trading partner. If you do not provide any certificate information, then these business documents cannot be signed, encrypted, or decrypted.



Note: The webMethods RosettaNet Module uses the same certificate information for signing and decrypting.

Security information is displayed on the **Security** tab.

Activating Your Trading Partners' Profiles

After you define a trading partner profile, you activate it by clicking Enable  .

Importing PIP Archives

■ What Is a PIP Archive?	38
■ Why Do I Need to Import a PIP Archive?	38
■ What Is In a PIP Archive?	38
■ PIPs and Process Models	39
■ Importing PIP Archives	39
■ Customizing PIP Archives	41
■ Exporting PIP Archives	42
■ Advantages and Disadvantages of Customizing and Exporting PIP Archives	45

What Is a PIP Archive?

A *PIP archive* is a compressed file (.par), also called a *process archive*, that contains webMethods RosettaNet Module configuration information and IS document types for a specific version of a particular PIP. PIP archives reside on the webMethods software download site at the URL provided to you by webMethods.

Each PIP archive has a file name that reflects the PIP number and version. For example, the file name for PIP 3A4 *Manage Purchase Order*, version 1.1, is PIP3A4v1_1.par.

Why Do I Need to Import a PIP Archive?

You need to import a PIP archive to implement a PIP in the webMethods RosettaNet Module. Because you download PIP archives from the webMethods software download site, you can add new PIP archives to your webMethods RosettaNet Module installation as they are released by webMethods, without having to upgrade your installation.

You can implement different versions of the same PIP to support transactions between you and your trading partners when not all of your trading partners are using the same version of the same PIP. However, in any specific PIP conversation, you and your trading partner must be using the same version of the PIP.

You cannot copy over the PIP archives for previous releases of the webMethods RosettaNet Module from one upgrade to the next. If you do not have customized PIPs and are upgrading to release 6.0.1, you should download and import the PIP archive. For customized PIPs, release 6.0.1 upgrade functionality will be added at a later date.

What Is In a PIP Archive?

A PIP archive contains all of the elements needed to implement the specific PIP. These elements include the following:

- **IS document types**, which define the namespace for the PIP and also define the structure of the RosettaNet PIP documents that are passed back and forth during the PIP conversation.
- **TN XML document types (internal and external)**, which the webMethods Integration Server uses to properly identify and route the PIP documents.
- **Trading Partner Agreements (TPAs)**, one for the initiator of a conversation and one for the fulfiller of a conversation.

PIPs and Process Models

Each RosettaNet PIP implements a business task or process. Each of these processes follows one of a few very basic process models. The differences between the specific PIP processes are mostly in the business documents that are passed back and forth and how those business documents are handled.

In [Chapter 2, “Concepts”](#), the basic overview of PIP transactions was presented using the example of the PIP 3A4 *Manage Purchase Order* process. This process is an example of a two-action asynchronous process model. In general, PIPs follow one of these three types of process models:

- **One-action asynchronous.** The initiating trading partner sends an asynchronous request, and the receiving trading partner sends a receipt acknowledgment. The webMethods RosettaNet Module contains two process model templates for the one-action asynchronous model: one template for the initiator and one template for the fulfiller. For information about these process model templates, see [Chapter 9, “Customizing a Process Model Template”](#) in this guide.
- **Two-action asynchronous.** The initiating trading partner sends an asynchronous request, and the receiving trading partner sends a receipt acknowledgment; the receiving partner sends an asynchronous response, and the initiating partner sends a receipt acknowledgment. The webMethods RosettaNet Module contains two process model templates for the two-action asynchronous model: one template for the initiator and one template for the fulfiller. For information about these process model templates, see [Chapter 9, “Customizing a Process Model Template”](#) in this guide.
- **Two-action synchronous.** The initiating trading partner sends a synchronous request, and the receiving partner sends a synchronous response over the same connection.

Importing PIP Archives

All PIP archives are located on the webMethods software download site at the URL provided to you by webMethods. You import a PIP archive by using the RosettaNet import facility in the Server Administrator.



Note: You should import PIP 0A1 *Notification of Failure* because this PIP is used by most other PIPs to communicate errors to your trading partners.


To import a PIP archive into the Integration Server

- 1 Navigate via a Web browser to the webMethods software download site at the URL provided to you by webMethods.
- 2 Enter your user name and password.

- 3 Scroll to the **webMethods RosettaNet Module** section on the page.
- 4 Click the PIP number and version that you want to import. For example, if you want to import PIP 3A4 *Manage Purchase Order*, version 1.1, click **PIP3A4 – 1.1**.
- 5 Save the file to the import subdirectory of your webMethods RosettaNet Module installation:

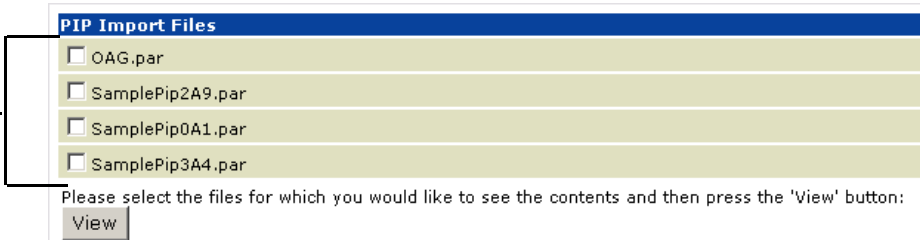
```
webMethods6\IntegrationServer\packages\WmRosettaNet\import
```

 where webMethods6\IntegrationServer is the directory in which your Integration Server is installed.
- 6 After saving the file to your file system, start the Integration Server and Server Administrator, if they are not already running.

 **Note:** If you need procedures for this step, see the *webMethods Integration Server Administrator's Guide*.

- 7 On the Server Administrator navigation panel, under the **Adapters** heading, click **RosettaNet**.
- 8 Click **PIP Import** to display the following page.

PIP Import Files Page




PIP files saved in the import subdirectory of your webMethods RosettaNet Module installation.

PIP Import Files	
<input type="checkbox"/>	OAG.par
<input type="checkbox"/>	SamplePip2A9.par
<input type="checkbox"/>	SamplePip0A1.par
<input type="checkbox"/>	SamplePip3A4.par

Please select the files for which you would like to see the contents and then press the 'View' button:

- 9 Select the check box next to the PIP you want to import.
- 10 Click **View**.
- 11 Scroll to the bottom of the page.

 **Important!** If an exclamation point appears to the left of the entries in the **Archive Contents** section of the **PIP Import File Details** table, then you have already installed this version of this PIP. If you want to use the version in the archive you are importing, continue this procedure.

If you have custom changes to PIP IS document types that you want to preserve, **stop** this procedure and follow the procedure in the [“Exporting PIP Archives” on page 42](#) in this guide. After exporting the customized PIP, continue with the next step in this procedure.

12 Click Import Archives.

Customizing PIP Archives

When implementing your webMethods RosettaNet Module installation, you might make numerous changes to the PIPs that you import. For example, you might develop your own custom TN XML document types that you and your trading partners agree to support. Or you might make changes to the IS document types against which you validate documents. The next two sections discuss two different approaches for customizing a PIP.



Note: For the advantages and disadvantages of customizing and exporting a PIP, see [“Advantages and Disadvantages of Customizing and Exporting PIP Archives”](#) on page 45 in this guide.

Customizing IS document types for Existing Standard RosettaNet PIPs

There are two ways to customize an IS document type for an existing standard RosettaNet PIP:

- If your trading partners are unable to populate and/or have no need for certain required fields in a PIP, you can change the constraints in those fields and make them optional.

Advantage: This type of customization enables your RosettaNet PIP documents for that PIP to pass validation.

Disadvantage: This type of customization affects every trading partner that will be exchanging this PIP with you.

- If you intend to isolate changes to an IS document type for a specific trading partner, follow these steps:
 - a Create another folder in the WmRNPips package under `wm.b2b.m.rec.PIPs` and name it *<your customized PIP structure>*.
 - b Enter the new location as the value for the *NSFolder* parameter in the TPA for that PIP.

Advantage: This type of customization localizes changes for one trading partner and for one particular PIP only.

Disadvantage: This type of customization provides you with one additional IS document type to maintain.

Implementing a Custom PIP

If you would like to implement a custom PIP other than the standard PIPs published by RosettaNet, follow the steps below.

To implement a custom PIP

- 1 In the Trading Networks Console, create a new TN XML document type and specify a root tag for the TN XML document type that matches the root tag of the customized PIP.
- 2 Add a pipeline matching variable to this new TN XML document type called *processVersion* and specify a value for this variable that matches the version in the Service Header of the customized PIP.
- 3 In the Server Administrator, create a new IS document type for the customized PIP.
- 4 In the Trading Networks Console, create a new Trading Partner Agreement (TPA). Specify the new TN XML document type as the **Agreement ID** for the TPA, and specify the new IS document type as the value for the *NSFolder* parameter in the TPA.
- 5 In webMethods Modeler, in the process model template for the PIP you are implementing, specify the new TN XML document type as input to the appropriate wait step.

Exporting PIP Archives

After you make changes to a PIP, you have essentially created a customized PIP from a standard PIP. Yet, if you simply import the new version of the PIP on top of the old version, your customizations will almost certainly be overwritten.

To avoid losing your changes the next time the RosettaNet organization or webMethods releases a new version of the PIP upon which you based your custom PIP, export the customized PIP and give it a different name than the standard PIP.

To export a PIP archive from the Integration Server

- 1 Start the Server Administrator, if it is not already running. If you need procedures for this step, see the *webMethods Integration Server Administrator's Guide*.
- 2 On the Server Administrator navigation panel, under the **Adapters** section, click **RosettaNet**.
- 3 On the Server Administrator navigation panel, under the **RosettaNet** section, click **PIP Export** to display the following page.

Create PIP Export Archive Page

Create PIP Export Archive	
File Name	<input type="text"/>
PIP Name	<input type="text"/>
PIP Number	<input type="text"/>
PIP Version	<input type="text"/>
Select TN Biz Doc Types to Export	<input type="checkbox"/> AcceptanceAcknowledgement
	<input type="checkbox"/> AcceptanceAcknowledgementException
	<input type="checkbox"/> Exception (RNIF 1.1)
	<input type="checkbox"/> Exception (RNIF 2.0)
	<input type="checkbox"/> Internal PO Acceptance
	<input type="checkbox"/> Internal PO Request
	<input type="checkbox"/> PIP3A4 v1.1 Purchase Order Acceptance Action Document
	<input type="checkbox"/> PIP3A4 v1.1 Purchase Order Request Action Document
	<input type="checkbox"/> PROFVAL_Profile
	<input type="checkbox"/> PROFVAL_extFields
	<input type="checkbox"/> PROFVAL_stdFields
	<input type="checkbox"/> RN11BizDocType
	<input type="checkbox"/> RN20BizDocType
<input type="checkbox"/> ReceiptAcknowledgement (RNIF 1.1)	
<input type="checkbox"/> ReceiptAcknowledgementException	
<input type="checkbox"/> ReceiptAcknowledgment (RNIF 2.0)	
<input type="checkbox"/> RosettaNet Dummy DocType	
<input type="checkbox"/> Unknown	
Select TPAs' to Export	<input type="checkbox"/> TPA between sender (") and receiver (") with AgreementID 'PIP3A4 v1.1 Purchase Order Request Action Document'
	<input type="checkbox"/> TPA between sender (") and receiver (") with AgreementID 'Internal PO Request'
Select PIP Folders to Export	<input type="checkbox"/> wm.b2b.rn.rec.PIPs.PIP3A4v1_1
Create Archive	

TN XML document types for the PIP.

Initiator and fulfiller Trading Partner Agreements (TPAs) for the PIP.

IS document types for the PIP.

- 4 Enter a file name for the new PIP archive file; for example, `PIP3Z4v1_1.par` for a custom PIP based on PIP 3A4.
- 5 Enter a PIP name for the new PIP; for example, `Manage Custom Purchase Order` for a custom PIP based on PIP 3A4 *Manage Purchase Order*.
- 6 Enter a PIP number for the new PIP; for example, `3Z4` for a custom PIP based on PIP 3A4.
- 7 Enter a PIP version number for the new PIP; for example, `1.1` for a custom PIP based on PIP 3A4 v 1.1.

In general, you should use the same version number as the standard PIP on which you based the custom PIP, at least for the initial creation of the custom PIP. Subsequent updates of the custom PIP might require a version numbering scheme that diverges from that of the standard PIP.

- 8 Select the check boxes for all of the TN XML document types that you want to export as part of the new PIP.

You should typically select all of the TN XML document types that were included in the standard PIP upon which you based your custom PIP, as well as any new TN XML document types you created for the custom PIP.

- 9 Select the check boxes for the Trading Partner Agreements (TPAs) that you want to export.
- 10 Select the check boxes for all of the PIP IS folders that you want to export as part of the new PIP.

You should typically select *only* the PIP IS folders for the standard PIP on which you based your customizations.

- 11 Click **Create Archive**.

Advantages and Disadvantages of Customizing and Exporting PIP Archives

If you decide to customize and export PIP archives, you should be aware of the advantages and disadvantages, which are listed in the following table.

Advantages of Customizing and Exporting	Disadvantages of Customizing and Exporting
<p>Enables the webMethods RosettaNet Module to manage your custom PIP as distinct from the standard PIP on which you based your customizations.</p>	<p>After you create a custom PIP based on a standard PIP, you must, from that point on, manually keep the custom process in parallel with the standard process; for example, when new features are added to the process.</p>
<p>Enables you to send the custom PIP to your trading partners (if they are also using the webMethods RosettaNet Module).</p>	<p>Customizing and exporting a PIP does not save any services or process models associated with that PIP. Exporting a PIP saves only the IS document types and TN XML document types associated with the PIP (and only those IS document types and TN XML document types that are contained in the <code>wm.b2b.m.rec.PIP</code> folder).</p> <p>To save the services, you must create your own package on the Integration Server and place all of your services within that package. You can then manage that package, as required by your installation, including sending that package to your trading partners. To save the process model for the customized PIP, you must use Modeler.</p>
<p>Enables you to do your own version management on your customized PIPs.</p>	

Defining Internal TN XML Document Types

- What are TN XML Document Types? 48
- TN XML Document Types Provided by WebMethods 48
- Defining Your Own Internal TN XML Document Types 49
- XML Queries Used to Extract Attributes From Business Documents 51

What are TN XML Document Types?

TN XML document types are definitions that tell webMethods Trading Networks how to identify a type of business document and specify the attributes that Trading Networks is to extract from the business document.

When the webMethods RosettaNet Module receives a business document, it invokes a Trading Networks service to recognize the type of business document by using the TN XML document types that were included in the respective PIP archive. When Trading Networks recognizes the TN XML document type of the business document, Trading Networks extracts specific pieces of information from the business document based on the document attributes specified in the TN XML document type.

For more information about TN XML document types, see the *Building Your Trading Network* manual.

TN XML Document Types Provided by WebMethods

When you install the webMethods RosettaNet Module on the webMethods Integration Server, the TN XML document types listed in the following table are defined in Trading Networks:

TN XML document type	Description
Acceptance Acknowledgement	XMLDocType for RNIF 1.1 AcceptanceAcknowledgement
AcceptanceAcknowledgementException	XMLDocType for RNIF 1.1 AcceptanceAcknowledgementException
Exception(RNIF1.1)	XMLDocType for RNIF 1.1 Exception
Exception (RNIF 2.0)	XMLDocType for RNIF 2.0 Exception
RN11BizDocType	RN11BizDocType for version 1.1
RN20BizDocType	RN20BizDocType for version 2.0
ReceiptAcknowledgement (RNIF 1.1)	XMLDocType for RNIF 1.1 ReceiptAcknowledgement
ReceiptAcknowledgementException	XML document type for RNIF 1.1 ReceiptAcknowledgementException
ReceiptAcknowledgement (RNIF 2.0)	XMLDocType for RNIF 2.0 ReceiptAcknowledgement
RosettaNet Dummy Document Type	Dummy document type used in the RosettaNet process model templates

You can view these TN XML document types in the Trading Networks Console.

Defining Your Own Internal TN XML Document Types

When you import a PIP archive, the internal and external TN XML document types for the PIP archive are automatically created in Trading Networks. However, there may be times that you will want to create your own internal TN XML document types or customize one of the provided internal TN XML document types.



Important! Do not modify any of the provided external TN XML document types. If you do, the incoming business document will not join the conversation.

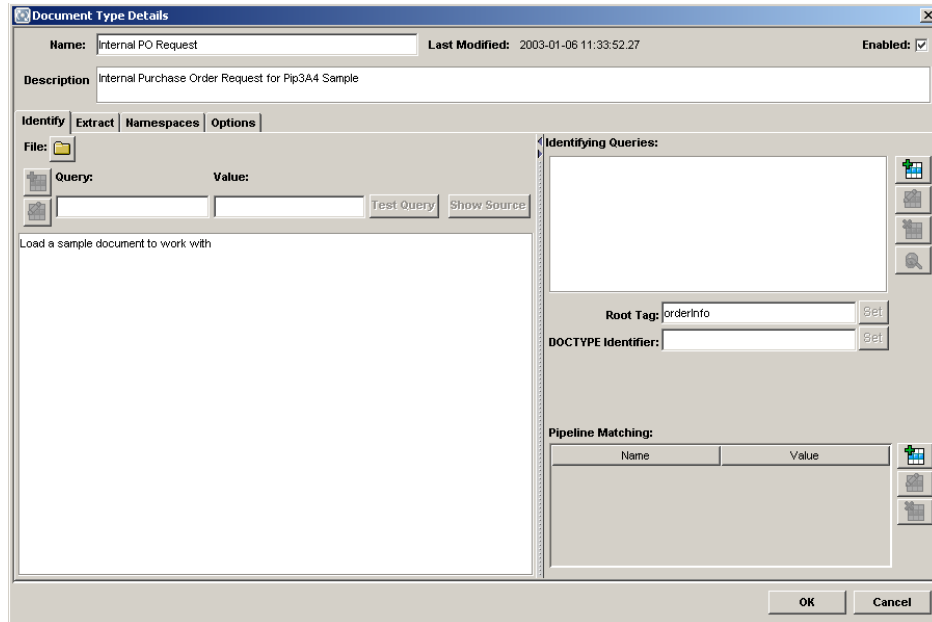
When you define an internal TN XML document type, you specify the root tag from within the business document that the TN XML document type is to match and the XML queries Trading Networks uses to extract the SenderID and ReceiverID attributes from the business document.

To define an internal TN XML document type


- 1 Start the Trading Networks Console. If you need procedures for this step, see the *webMethods Trading Networks--Getting Started with Trading Networks* manual.
- 2 In the Selector Panel, click **My Enterprise**.
- 3 Select **View** ▶ **Document Types**.
- 4 Select **Types** ▶ **New**.
- 5 In the **Create New DocType** dialog box, select the **XML** document type category from the list, and click **OK**.
- 6 In the **Document Type Details** screen, the **Name** field, type the name you want to give the internal TN XML document type.
- 7 In the **Description** field, type a description for the internal TN XML document type.
- 8 In the **Root Tag** field, type the value of the root tag from within the business documents that this internal TN XML document type is to match.

The following figure illustrates the **Document Type Details** screen with the aforementioned three fields completed.

Document Type Details Screen for an Internal TN XML Document Type



Note: You can tell if a TN XML document type is an external TN XML document type by the description in the **Description** field in the **Document Type Details** screen and also by the fact that an external TN XML document type *always* has a pipeline matching variable of *processVersion*. That the above TN XML document type is an internal TN XML document type is readily apparent because there is *no* pipeline matching variable.

- 9 Specify SenderID as one of the attributes to extract.
 - a Click the **Extract** tab.
 - b Click **Add an Attribute**  .
 - c In the **Add an Attribute** dialog box, the **Name** list, select **SenderID**.
 - d Select the **Required** check box.
 - e In the **Description** field, enter a description for the attribute.
 - f In the **Query** field, enter the query to extract the SenderID attribute. For the query to extract the SenderID attribute, see [“SenderID XML Query” on page 51](#) in this guide.
 - g In the **Detail** list, select **DUNS**.
 - h Click **OK**.
- 10 Repeat step 9 for the ReceiverID attribute. For the query to extract the ReceiverID attribute, see [“ReceiverID XML Query” on page 51](#) in this guide.

XML Queries Used to Extract Attributes From Business Documents

Trading Networks uses XML queries to extract the SenderID and ReceiverID attributes from internal business documents sent to the webMethods RosettaNet Module from a back-end system. Trading Networks uses the SenderID and ReceiverID to determine who is sending the business document and whom to send the business document to.



Note: If a business document is an external business document received from a trading partner, Trading Networks uses the SenderID and ReceiverID from the business document's Service Header.

The following sections provide the XML queries used to extract the SenderID and Receiver ID attributes.

SenderID XML Query

The SenderID is always the D-U-N-S[®] Number of the sender. The query to extract the SenderID for business documents (for example, Order Create, OrderResponse, and so on) is the following:

```
/ServiceHeader [0]/ProcessControl [0]/TransactionControl
[0]/ActionControl/PartnerRoute [0]/fromPartner [0]/PartnerDescription
[0]/BusinessDescription [0]/GlobalBusinessIdentifier [0]
```

The query to extract the SenderID for signal documents (for example, Receipt ACK, Exception, and so on) is as follows:

```
/ServiceHeader [0]/ProcessControl [0]/TransactionControl
[0]/SignalControl [0]/PartnerRoute [0]/fromPartner [0]/PartnerDescription
[0]/BusinessDescription [0]/GlobalBusinessIdentifier [0]
```

ReceiverID XML Query

The ReceiverID is always the D-U-N-S Number of the receiver. The query to extract the ReceiverID for business documents (for example, OrderCreate, OrderResponse, and so on) is as follows:

```
/ServiceHeader [0]/ProcessControl [0]/TransactionControl
[0]/ActionControl [0]/PartnerRoute [0]/toPartner [0]/PartnerDescription
[0]/BusinessDescription [0]/GlobalBusinessIdentifier [0]
```

The query to extract the ReceiverID for signal documents (for example, Receipt ACK, Exception, and so on) is as follows:

```
/ServiceHeader [0]/ProcessControl [0]/TransactionControl  
[0]/SignalControl [0]/PartnerRoute [0]/toPartner [0]/PartnerDescription  
[0]/BusinessDescription[0]/GlobalBusinessIdentifier [0]
```

Customizing Trading Partner Agreements

■ What is a Trading Partner Agreement?	54
■ How Does the webMethods RosettaNet Module Identify a TPA?	54
■ Modifying the Sender, Receiver, and Agreement ID	54
■ Distinguishing Between an Initiator's TPA and a Fulfiller's TPA	57
■ Parameter Settings	57

What is a Trading Partner Agreement?

A *Trading Partner Agreement (TPA)* is a set of parameters that you can use to govern how business documents are exchanged between two trading partners. When you import a PIP archive (.par file) into the webMethods Integration Server, the TPAs for that PIP are created in webMethods Trading Networks. You view and customize the TPAs in the Trading Networks Console **Agreement Details** screen. For information about working with TPAs in the Trading Networks Console, see the *Building Your Trading Network* manual.

Every PIP in the webMethods RosettaNet Module is associated with two TPAs, one for the initiator/sender in a conversation and one for the fulfiller/receiver in a conversation. The TPAs provided with the webMethods RosettaNet Module contain a set of parameters that map to some (but not all) elements in the Service Header of a business document. The TPA parameters and the Service Header elements that the parameters map to vary between RNIF version 1.1 and 2.0.

How Does the webMethods RosettaNet Module Identify a TPA?

Every TPA is uniquely identified by a Sender, Receiver, and Agreement ID. During a conversation between trading partners, the webMethods RosettaNet Module uses this information to retrieve the TPAs for the initiator and fulfiller in the conversation and to process the business documents exchanged.

The Agreement ID is *always* the first TN XML document type name used to start the PIP process. For the initiator of a PIP conversation, the first TN XML document type name is typically the TN XML document type name representing the internal back-end document. For example, in the PIP 3A4 sample, the Agreement ID of the TPA for the “Buyer” conversation is “Internal PO Request.” For the fulfiller of a PIP conversation, the first TN XML document type name is typically the TN XML document type name of the first RosettaNet PIP document received. For example, in the PIP 3A4 sample, the Agreement ID of the TPA for the “Seller” conversation is “Pip3A4 v1.1 Purchase Order Request Action Document.”

Modifying the Sender, Receiver, and Agreement ID

When a TPA is created in Trading Networks, it is presented in the **Agreement Details** screen. In the **Agreement Details** screen, the **Sender** and **Receiver** fields (for both the initiator’s and fulfiller’s TPAs) initially display a default value of “Unknown”, as illustrated in the following figure.

Agreement Details Screen

Sender, Receiver, and Agreement ID

Double-click a heading to view the parameters that fall under that heading.

Modifying an Initiator's TPA

In an initiator's TPA, you modify the Sender, Receiver, and Agreement ID, which is a placeholder for the TN XML document type name of the internal business document received from the back-end system. In the figure in the preceding section, the Agreement ID placeholder of "Internal PO Request" indicates that the TPA is for an initiator because the starting business document for an initiator's TPA is always an internal business document that is from a back-end system.



Important! The webMethods RosettaNet Module does **not** support a TPA in which either the Sender is known (for example *Company1*) and the Receiver is *Unknown* or in which the Sender is *Unknown* and the Receiver is known (for example, *Company2*). The webMethods RosettaNet Module supports only those TPAs in which either both the Sender and Receiver are known (for example, *Company1* and *Company2*, respectively) or in which the Sender and Receiver are *Unknown*.

Modifying a Fulfiller's TPA

In a fulfiller's TPA, you modify, the Sender and Receiver *only*. You must **not** modify the Agreement ID. After the webMethods RosettaNet Module recognizes and associates an incoming RosettaNet object (RNO) to a particular TN XML document type, it uses the TN XML document type name in the bizdoc to find a matching TPA. If you were to change the Agreement ID, the webMethods RosettaNet Module could no longer match the TN XML document type name to the Agreement ID.



Important! The webMethods RosettaNet Module does **not** support a TPA in which either the Sender is known (for example *Company1*) and the Receiver is *Unknown* or in which the Sender is *Unknown* and the Receiver is known (for example, *Company2*). The webMethods RosettaNet Module supports only those TPAs in which either both the Sender and Receiver are known (for example, *Company1* and *Company2*, respectively) or in which the Sender and Receiver are *Unknown*.

Field Descriptions

The following table lists and describes the TPA information that you need to understand at the top of the **Agreement Details** screen.

TPA Information	Description
Sender	The name of the trading partner that has the sender role in the TPA. You select the sender from the profiles defined on your Trading Networks system, including your own profile (My Enterprise). The Sender is always a D-U-N-S® Number.
Receiver	The name of the trading partner that has the receiver role in the TPA. You select the receiver from the profiles defined on your Trading Networks system, including your own profile (My Enterprise). The Receiver is always a D-U-N-S Number.
Agreement ID	An application-specific field that uniquely identifies the type of agreement between two partners.
IS Document Type	The <code>wm.ip.mn.rec:UserParameters</code> IS document type that specifies the data, or the parameters, that you define in the TPA. You can find the <code>wm.ip.mn.rec:UserParameters</code> IS document type in the <code>WmRosettaNet</code> package. To define a TPA, you need to know the fully qualified name of this IS document type.
Data Status	This field applies only when the Agreement Status is Agreed . The data status indicates whether you can update the values for the TPA parameters defined in the IS document type (located in the panel on the lower portion of the Agreement Details screen). The data status can be one of the following: <ul style="list-style-type: none"> ■ Modifiable - the TPA data can be changed ■ Non-modifiable - the TPA data cannot be changed

For step-by-step instructions about how to modify a TPA, see the *Building Your Trading Network* manual.

Distinguishing Between an Initiator's TPA and a Fulfiller's TPA

You distinguish between an initiator's TPA and a fulfiller's TPA by the Agreement ID. If the TPA in the figure in [“Modifying the Sender, Receiver, and Agreement ID”](#) on page 54 in this guide were for the fulfiller in the conversation, the Agreement ID might be PIP3A4v1.1 Purchase Order Request Action Document, which is the TN XML document type name that specifies a RosettaNet PIP document for PIP 3A4 *Manage Purchase Order*. The Agreement ID for the fulfiller's TPA always includes the PIP name.

You might want to delete the TPA that you do not use. For example, if you are the initiator in a conversation, you would delete the fulfiller's TPA. If you are the fulfiller in a conversation, you would delete the initiator's TPA.

For a list and description of the TPA parameters, see the next section, [“Parameter Settings”](#).

Parameter Settings

TPA parameters are divided into three sections:

- **ProcessInfo.** Consists of process-level information, such as the Transport, the TransportVersion, and so on.
- **PipInfo.** Consists of information about the PIP process, such as the DTD, ActionCode, and so on. Do **not** edit PIP information unless you are implementing a customized PIP.
- **RNIF2.0.** Consists of information specifically for RNIF 2.0 processes, such as Encoding, EncryptPayload, and so on.

The following table lists and describes the TPA parameters for an RNIF 1.1 TPA and an RNIF 2.0 TPA.

TPA Section	Parameter	Description
ProcessInfo	Transport	The transport protocol type; for example, RNIF.
	TransportVersion	The version of the transport protocol; that is, 1.1 or 2.0.
	Sign	Whether the outbound RosettaNet PIP document for the process model should be signed. Valid values are <i>Yes</i> or <i>No</i> .
	SignatureRequired	Whether the inbound RosettaNet PIP document for the process model should contain your trading partner's signature information. Valid values are <i>yes</i> or <i>no</i> .
	HashAlgorithm	The algorithm to use to generate the RosettaNet Message Digest; that is, MD5 or SHA-1.

TPA Section	Parameter	Description
ProcessInfo (cont.)	UseProfileCerts	For HTTPS connections, whether to use the certificates from the Sign tab in the Security section of the receiving trading partner profile. The default value for this parameter is <i>No</i> , indicating that the webMethods RosettaNet Module will use the Integration Server default certificate. <i>Yes</i> indicates that the webMethods RosettaNet Module will use the certificates from the receiving trading partner profile.
	ValidationService	The service that validates the RosettaNet PIP document. This parameter is used for large documents only. Note: If your process model handles both large and small business documents, then your validation service must handle both large and small business documents.
	ValidateOutput	Whether validation of the RosettaNet PIP document will be performed before you send the business document to your trading partner. Valid values are <i>Yes</i> or <i>No</i> .
PipInfo	ProcessCode	The code for, or name of, the PIP, such as PIP3A4. The value for this parameter is derived from the PIP Specification. The value for this parameter applies to a particular PIP and a particular version of a PIP.
	ProcessVersion	The version of the PIP process, such as 1.1 or 1.4. The value for this parameter is derived from the PIP Specification. The value for this parameter applies to a particular PIP and a particular version of a PIP.
	ProcessDescription	The description of the PIP process; for example, the description of PIP3A4 is Manage Purchase Order. The value for this parameter is derived from the PIP Specification. The value for this parameter applies to a particular PIP and a particular version of a PIP.
	TransactionCode	The transaction associated with each RosettaNet PIP document. The value for this parameter is derived from the PIP Specification. The value for this parameter applies to a particular PIP and a particular version of a PIP.

TPA Section	Parameter	Description
PipInfo (cont.)	ActionCode	The action associated with each RosettaNet PIP document. The value for this parameter is derived from the PIP Specification. The value for this parameter applies to a particular PIP and a particular version of a PIP.
	DTD	The file name of the Data Type Definition for the RosettaNet PIP document. The value for this parameter is derived from the PIP Specification. The value for this parameter applies to a particular PIP and a particular version of a PIP.
	NSFolder	The folder that contains the IS document types against which the RosettaNet PIP documents will be validated. The value for this parameter applies to a particular PIP and a particular version of a PIP.
	UsageCode	Whether the process model is in Test or Production mode. Valid values are <i>Test</i> or <i>Production</i> .
	SenderClassification	The business classification of the sender; for example, Manufacturer, Distributor, Customers Broker.
	ReceiverClassification	The business classification of the receiver.
	RNIF2.0	Encoding
EncryptPayload		Whether you want the outbound RosettaNet PIP document to be encrypted. Valid values are <i>Yes</i> or <i>No</i> .
EncryptHeader		Whether you want the Service Header of the RosettaNet PIP document to be encrypted, along with any attachments, if present. Valid values are <i>Yes</i> or <i>No</i> .
EncryptionAlgorithm		The algorithm used to encrypt the outbound RosettaNet PIP document. Valid values are <i>Triple DES</i> , <i>RC2</i> , and <i>DES</i> .
RC2EncryptionKeyLength		The length (in bits) of the RC2 encryption key. Valid values are <i>128</i> , <i>64</i> , and <i>40</i> .
ThirdPartyPayloadType		The payload type for the third-party specification. webMethods supports only Open Applications Group (OAG) as the third party. This parameter is equivalent to the <i>ProcessCode</i> parameter for PIPs.

TPA Section	Parameter	Description
RNIF2.0 (cont.)	ThirdPartyPayloadVersion	The payload version for the third-party specification. webMethods supports only OAG as the third party. This parameter is equivalent to the <i>ProcessVersion</i> parameter for PIPs.
	ThirdPartyReferenceId	The payload reference ID for the third-party specification. webMethods supports only (OAG) as the third party.

Mapping a RosettaNet PIP Document

■ What Is “Mapping” a Business Document?	62
■ Creating an Outbound Mapping Service	64
■ Creating an Inbound Mapping Service	67
■ Reusing Mapping Services	70

What Is “Mapping” a Business Document?

“Mapping” a business document refers to the process of assigning the structure, values, or content of one or more business documents to a new document; that is, mapping the values, data, and information from various sources into a single, new business document. You need to map business documents because, typically, your back-end systems have different document formats than that of the RosettaNet PIP documents.



Note: For information about mapping large documents, see [Appendix A, “Processing Large Business Documents”](#), in this guide.

Why Do You Create an Outbound Mapping Service?

You create an outbound mapping service to translate an outbound back-end proprietary business document (for example, an IDOC from an SAP R/3 system) to a RosettaNet PIP document. Elements of the proprietary business document need to be mapped to corresponding elements in a specific version of a RosettaNet PIP (for example, PIP 3A4 *Manage Purchase Order*, version 1.1) document. Extra elements in the back-end business document are ignored, as long as values are mapped to **all** elements in the RosettaNet PIP document. Examples of outbound mapping services are located in the WmRNSample package.

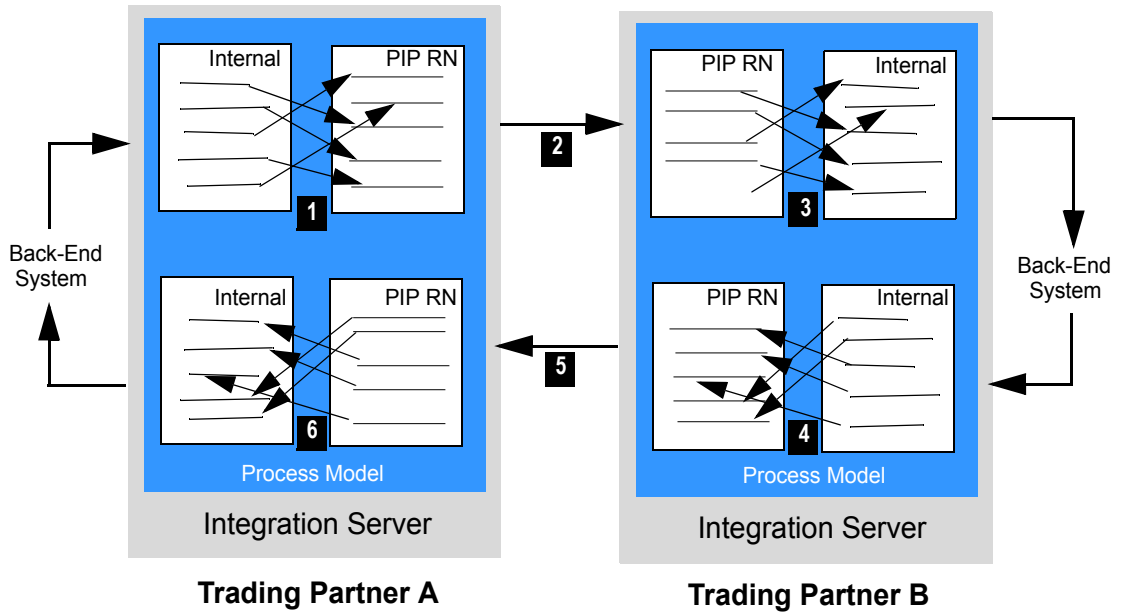
Why Do You Create an Inbound Mapping Service?

You create an inbound mapping service to map each element of the inbound RosettaNet PIP document to a corresponding element in your back-end proprietary business document format. For example, if you use SAP R/3 as your back-end ERP system, and you want to receive a RosettaNet PIP document via the webMethods RosettaNet Module, you create an inbound mapping service that maps each element of the RosettaNet PIP document to a corresponding element in the IDOC format. Examples of inbound mapping services are located in the WmRNSample package.

Example of Mapping a Business Document

The following figure illustrates the process of mapping a business document for a two-action PIP. For further explanation, see the text that follows the figure.

Example of Mapping a Business Document



Step	Description
1	Trading Partner A uses an outbound mapping service to map an internal business document from a back-end system to a RosettaNet PIP document.
2	Trading Partner A sends the RosettaNet PIP document to Trading Partner B.
3	Trading Partner B receives the RosettaNet PIP document and uses an inbound mapping service to map the RosettaNet PIP document to an internal business document. After the internal business document is mapped, it is in a format that Trading Partner B's back-end system can process.
4	Trading Partner B responds by using an outbound mapping service to map an internal business document from the back-end system to a RosettaNet PIP document.

Step	Description
5	Trading Partner B sends the RosettaNet PIP document to Trading Partner A.
6	Trading Partner A receives the RosettaNet PIP document and uses an inbound mapping service to map the RosettaNet PIP document to an internal business document. After the business document is mapped, it is in a format that Trading Partner A's back-end system can process.

Creating an Outbound Mapping Service

In webMethods Developer, you create an outbound mapping service by creating a new flow service. This flow service contains one or more MAP entities, which do the actual mapping from your back-end business document, through any desired intermediate steps, to the IS document type for the appropriate outbound RosettaNet PIP document.



Note: For information about customizing your outbound mapping service to handle large documents, see [Appendix A, “Processing Large Business Documents”](#), in this guide.

Input/Output to Use

The input to the outbound mapping service is, indirectly, your back-end business document, represented as an IData object.



Note: You typically invoke `wm.ip.util:getBizDocFromEvent` to retrieve the `BizDocEnvelope`, which contains your back-end business document, and then invoke `wm.ip.util:recordFromBizDoc` to extract the business document content from the `BizDocEnvelope`. This process returns your back-end business document as an IData object.

The output from the outbound mapping service is the RosettaNet PIP document in the **documents\Payload** IData object in the pipeline. For instance, if you are mapping your back-end documents to the *PIP3A4PurchaseOrderRequest* RosettaNet PIP document, the output of the mapping service would be *PIP3A4PurchaseOrderRequest(PIP3A4PurchaseOrderRequest)* in the **documents\Payload** IData object. See the figure in “[Example of an Outbound Mapping Service: Initiator/Sender](#)” on [page 66](#) in this guide.

The output from the outbound mapping service **must** be placed in the **documents\Payload** IData object. The webMethods RosettaNet Module has to convert a PIP IData object into an XML string before sending the RosettaNet PIP document to the trading partner.

Therefore, the webMethods RosettaNet Module must know the precise location of the PIP IData object.

Note: If the documents for your back-end system have DTDs, you can automatically import an external DTD in Developer to provide a starting point for mapping. Simply create a new external record and specify the source as *XML*, *DTD*, or *XML Schema*.

Flow Operations to Use

In the flow service, you insert a MAP operation and use the service pipeline to map elements of the IS document type for your back-end business document to all elements of the IS document type for the appropriate RosettaNet PIP document. Built-in IS document types for all versions of the RosettaNet PIPs that you imported are located in the WmRNPips package in the wm.b2b.m.rec.PIPs folder as illustrated in the following figure.

PIP Records Available for Mapping

The screenshot shows the webMethods Developer interface. On the left, a tree view displays the project structure under 'localhost:5555'. A folder named 'WmRNPips' is expanded, showing subfolders 'b2b', 'm', and 'rec'. The 'rec' folder is further expanded to show 'PIPs', which contains the record 'PIP3A4v1_1'. This record is expanded to show 'Pip3A4PurchaseOrderAcceptance' and 'Pip3A4PurchaseOrderRequest'. An arrow points from the text 'This folder contains all IS document types that you imported.' to the 'WmRNPips' folder.

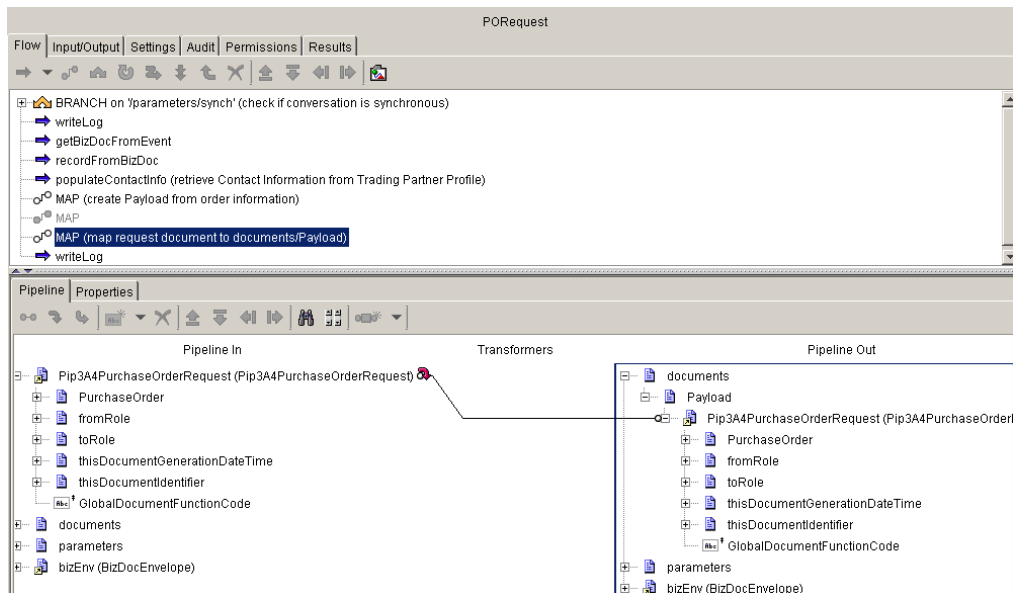
On the right, the 'Pip3A4PurchaseOrderRequest' record is selected, and its structure is displayed in a tree view. The structure includes the following elements:

- PurchaseOrder
 - deliverTo
 - comment
 - packListRequirements
 - ProductLineItem
 - GlobalShipmentTermsCode
 - RevisionNumber
 - prePaymentCheckNumber
 - QuoteIdentifier
 - WireTransferIdentifier
 - AccountDescription
 - generalServicesAdministrationNumber
 - secondaryBuyerPurchaseOrderIdentifier
 - GlobalFinanceTermsCode
 - PartnerDescription
 - secondaryBuyer
 - GlobalPurchaseOrderTypeCode
 - fromRole
 - toRole
 - thisDocumentGenerationDateTime
 - thisDocumentIdentifier
 - GlobalDocumentFunctionCode

Example of an Outbound Mapping Service: Initiator/Sender

If you are the initiator/sender in a conversation and are implementing a version of PIP 3A4 *Manage Purchase Order*, an outbound mapping service converts your back-end business document to a RosettaNet PIP document (PO Request). Such an outbound mapping service might look like the following figure. For further explanation, see the text that follows the figure.

Example of an Outbound Mapping Service: Initiator/Sender



Under the **Pipeline Out** heading, in the **documents\Payload** IData object, the **Pip3A4PurchaseOrderRequest** not in parentheses is the *IData object*, or RosettaNet PIP document, being sent to the trading partner and the **Pip3A4PurchaseOrderRequest** in parentheses is the *IS document type* that defines the RosettaNet PIP document. The name of the IData object is case sensitive.

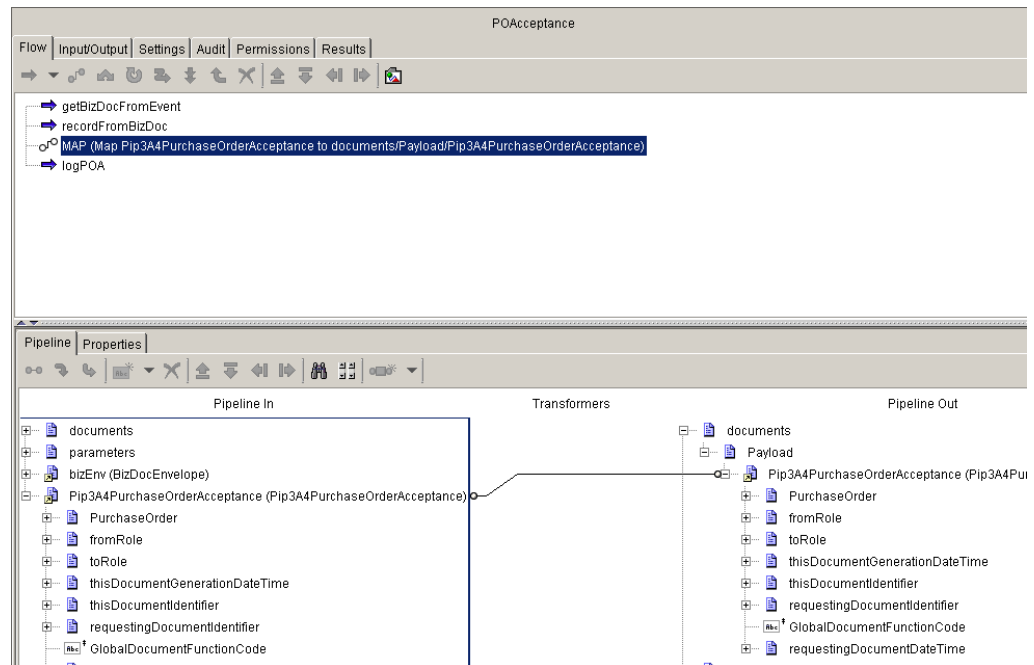


Important! As a general convention, the name of the IData object should be the same (including the case) as the name of the IS document type. If not, the validation on the receiver's end will fail. Using the above figure as an example, the receiver's RosettaNet-compliant server will expect the name of the incoming RosettaNet PIP document to have a RosettaNet name of **Pip3A4PurchaseOrderRequest**.

Example of an Outbound Mapping Service: Fulfiller/Receiver

If you are the fulfiller/receiver in a conversation and are implementing a version of PIP 3A4 *Manage Purchase Order*, the outbound mapping service might look like the following figure. For further explanation, see the text that follows the figure.

Example of an Outbound Mapping Service: Fulfiller/Receiver



The fulfiller returns a Purchase Order (PO) Acceptance to the initiator. Note that the **Pip3A4PurchaseOrderAcceptance** IData object that contains information from the fulfiller’s back-end system is mapped to the **Pip3A4PurchaseOrderAcceptance** IData object, or the RosettaNet PIP document in the **documents\Payload** IData object.

Creating an Inbound Mapping Service

In webMethods Developer, just as with outbound mapping services, you create an inbound mapping service by creating a new flow service. This flow service contains one or more MAP entities, which do the actual mapping from the received RosettaNet PIP document, through any intermediate steps, to the format of your back-end business documents.



Note: For information about customizing your inbound mapping service to handle large documents, see [Appendix A, “Processing Large Business Documents”](#), in this guide.

Input/Output to Use

For RNIF version 1.1, the inputs to the inbound mapping service are the following IData objects: **userDocument**, the Preamble Header, the Service Header, and Attachments (optional). For RNIF version 2.0, the inputs to the inbound mapping service are the following IData objects: **userDocument**, the Preamble Header, the Service Header, the Delivery Header, and Attachments (optional).

The **userDocument** IData object **must** be named **userDocument**, which is the IData object in which the webMethods RosettaNet Module places the inbound RosettaNet PIP document after validating it and recording it to the Integration Server Repository. By placing the inbound RosettaNet PIP document in the **userDocument** IData object, the webMethods RosettaNet Module makes the information immediately available without requiring an additional access call to data storage (thus improving performance). See the figure in [“Example of an Inbound Mapping Service: Initiator/Sender”](#) on page 69 in this guide.

For an example of inbound mapping, see the sample in the WmRNSample package. For instructions about using the sample, see the *webMethods RosettaNet Module Sample Guide*



Note: If the documents for your back-end system have DTDs, you can automatically import an external DTD in webMethods Developer to provide a starting point for mapping. Simply create a new external record and specify the source as *XML*, *DTD*, or *XML Schema*.

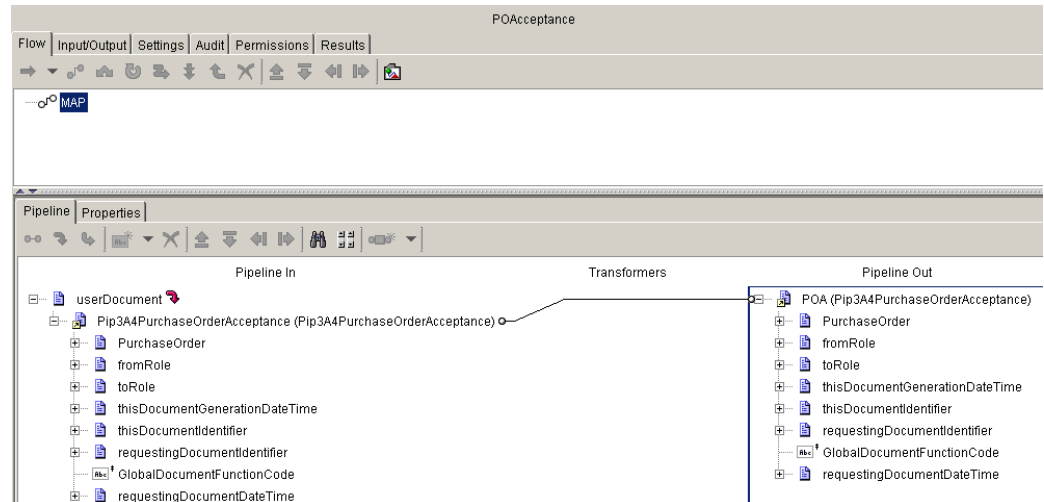
Flow Operations to Use

In the flow service, you insert a MAP operation and use the pipeline to map elements of the IS document type for the RosettaNet PIP document to elements in your back-end business document.

Example of an Inbound Mapping Service: Initiator/Sender

If you are the initiator/sender in a conversation and are implementing a version of PIP 3A4 *Manage Purchase Order*, the inbound mapping service might look like the following figure. For further explanation, see the text that follows the figure.

Example of an Inbound Mapping Service: Initiator/Sender

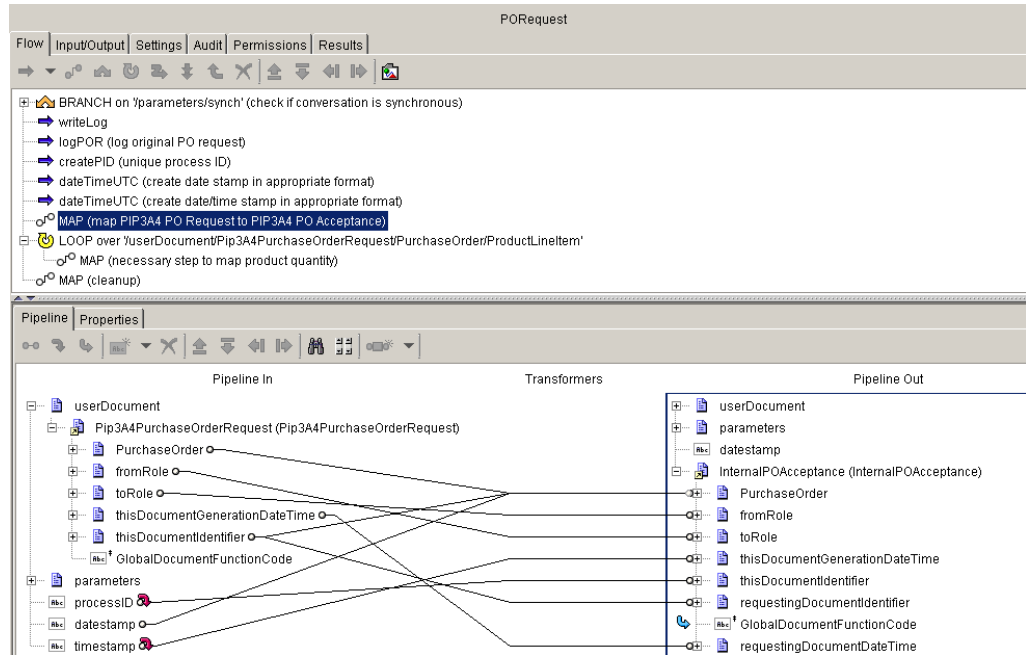


When the webMethods RosettaNet Module for the initiator receives the inbound PO Acceptance from the fulfiller, it places the **Pip3A4PurchaseOrderAcceptance** IData object in the **userDocument** IData object in the pipeline. The **Pip3A4PurchaseOrderAcceptance** IData object is mapped to the **POA** I Data object, a back-end business document.

Example of an Inbound Mapping Service: Fulfiller/Receiver

If you are the fulfiller/receiver in a conversation and are implementing a version of PIP 3A4 *Manage Purchase Order*, the inbound mapping service might look like the following figure. For further explanation, see the text that follows the figure.

Example of an Inbound Mapping Service: Fulfiller/Receiver



When the webMethods RosettaNet Module for the fulfiller receives the inbound PO from the initiator, it places the **Pip3A4PurchaseOrderRequest** IData object in the **userDocument** IData object in the pipeline. The **Pip3A4PurchaseOrderRequest** IData object is mapped to the **InternalPOAcceptance** IData object, a back-end business document.

Reusing Mapping Services

In the webMethods RosettaNet Module, you can reuse mapping services for trading partners that submit the same business document format. For example, you can use the same mapping services for Trading Partner A and Trading Partner B if they use the same version of the same PIP and they both always submit business documents in the same document format to the webMethods RosettaNet Module. As the receiver of those documents, you only need to define one inbound mapping service for both trading partners, because the document format and PIP versions are the same.

You generally cannot reuse the same mapping service for different versions of the same PIP because the mapping service includes references to IS document types within a specific version of the PIP.

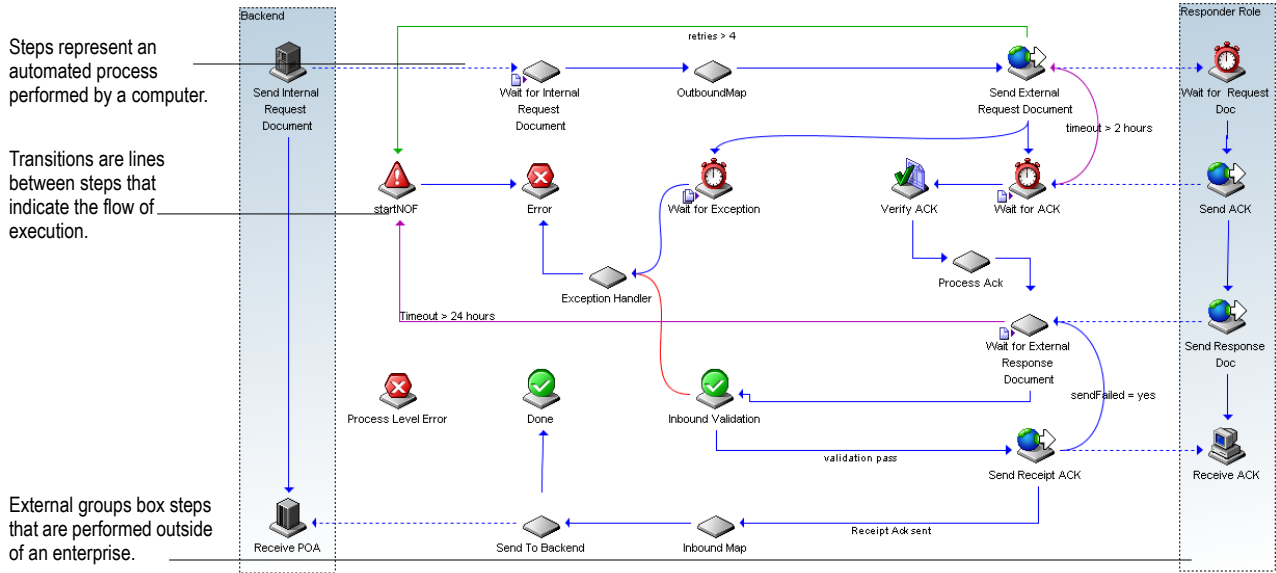
Customizing a Process Model Template

- What is a Process Model? 74
- Process Model Templates Provided by webMethods 76
- Steps in the Process Model Templates That You Must Modify 77
- Prerequisites for Customizing a Process Model Template 78
- Customizing a Process Model Template 78

What is a Process Model?

A process model is a diagram that represents a business process. The following figure illustrates a sample process model. For further explanation, see the text that follows the figure.

Sample Process Model



A process model consists of:

- **Steps.** The basic unit of work in a process model.
- **Transitions.** The lines between steps that indicate the execution order of steps within the process model.
- **Groups.** Clustering steps to represent different organizational boundaries within a process model.
- **Annotations (notes or text).** The labels, notes, and explanatory text in a process model.

For more information about steps, transitions, groups, and annotations, see the *webMethods Modeler User's Guide*.

The steps in a process model determine how the process run time (PRT) conducts a conversation, which includes how the process run time processes a RosettaNet PIP message. For information about the process run time, see "[webMethods RosettaNet Module Run-Time Architecture/Components](#)" on page 20 in this guide and the *webMethods Modeler User's Guide*.

webMethods provides process model templates that you can use to build your own process models. You customize the process model templates by specifying how the

process model is to interact with your back-end systems. You do this by editing the services that are invoked by the steps of the process model, replacing the RosettaNet Dummy Doc Types with the TN XML document types that you have created for steps that are waiting for action documents, specifying inbound and outbound mapping services, and enhancing the provided error and exception handling services, if necessary. When customizing a process model template, you also assign TN roles to TN XML document types and a focal role to the process model.

You use webMethods Modeler to customize the process model templates based on the PIPs you are implementing. For more information about process models and Modeler, see the *webMethods Modeler User's Guide*.

TN Roles and Focal Roles

When you customize a process model template, you assign both TN roles and a focal role.

TN Roles

In Modeler, you assign TN roles to TN XML document types that are associated with steps that wait to receive action documents, such as a purchase order (PO) or a PO acceptance (POA). A *TN role* specifies the role of a user for a specific business document. The TN roles that you can assign to a TN XML document type for an action document depend on the PIP you are implementing.

To find the TN roles to assign to TN XML document types, refer to the RosettaNet Web site (<http://www.rosettanet.org>), and look in the Partner Role Description table in the PIP Specification for the PIP you are implementing. For example, the TN roles you would assign to TN XML document types associated with steps that are waiting for either a PO or a POA in a PIP 3A4 *Manage Purchase Order* process model are “Buyer” and “Seller,” as indicated in the Partner Role Description table in the PIP 3A4 *Manage Purchase Order* Specification. If you send a PO to your trading partner, you, acting as a Buyer, are the initiator/sender for the PO and your trading partner, acting as the Seller, is the fulfiller/receiver of the PO. When you receive a POA from your trading partner, you, the Buyer, are the fulfiller/receiver of the POA, and your trading partner, the Seller, is the initiator/sender of the POA. In other words, you or your trading partner may be either the initiator/sender or the fulfiller/receiver for a business document depending on the document being sent, regardless of the role that you or your trading partner play in the conversation as a whole, as indicated in the following illustration.

TN Roles



When you replace a RosettaNet Dummy Doc Type with a TN XML document type that you created for a particular wait step, Modeler prompts you to enter the TN roles for the **Sender Role** and the **Receiver Role** in the **Set TN Roles** dialog box. For example, if you are replacing a RosettaNet Dummy Doc Type for a particular wait step with the name of a PO that you are sending to a trading partner, you would enter “Buyer” as the **Sender Role** and “Seller” as the **Receiver Role**.

Focal Role

When you customize a process model template, you also assign a focal role to the process model. A *focal role* specifies the role of the user for a particular conversation, or process. As with TN roles, you determine the focal role you need to use from the Partner Role Description table in the PIP Specification for the PIP you are implementing. You assign a focal role for your enterprise in the **Properties** dialog box in Modeler.

Process Model Templates Provided by webMethods

webMethods provides the following process model templates to assist you in creating your process models:


- One Action PIP Initiator Model.xml
- One Action PIP Fulfiller Model.xml
- Two Action PIP Initiator Model.xml
- Two Action PIP Fulfiller Model.xml

You can find the process model templates in the following location:

webMethods6\IntegrationServer\packages\WmRosettaNet\ProcessModels

where webMethods6\IntegrationServer is the directory in which the webMethods Integration Server is installed.

Steps in the Process Model Templates That You Must Modify

In the process model templates, steps that you must modify are denoted with a plain, diamond-shaped icon . The following table lists the steps in the process model templates that you must modify, as well as descriptions of the modifications and whether a step applies to the initiator, fulfiller, or both.

Process Model Template Step	Description of Modification	Initiator	Fulfiller
Wait for Internal Request Document	Add a subscription to an internal TN XML document type.	✓	
Wait for External Request Document	Add a subscription to an external TN XML document type.		✓
Outbound Map	Add an outbound mapping service to map your back-end document to a RosettaNet PIP document.	✓	✓
Inbound Map	Add an inbound mapping service to map your RosettaNet PIP document to a back-end document.	✓	✓
Send to Back-End	Add a service to send the business document to the back-end system.	✓	✓
Wait for External Response Document	Add a subscription to an external TN XML document type.	✓	
Wait for Internal Response Document	Add a subscription to an internal TN XML document type.		✓
Process Ack	Add a service to handle the acknowledgement; for example send the ACK to another department in your enterprise, trigger another service, and so on.	✓	✓
Exception Handler	Add a service to handle the exception condition in your desired way; for example, to send an e-mail message.	✓	✓



Note: The process model templates contain one step that you can modify if you want to. This step is the **Error** step. You can modify this step only if you will be invoking `wm.ip.cm.handlers:error` in the service that you are going to specify for this step.

Prerequisites for Customizing a Process Model Template

The following table lists the prerequisites you should complete before you customize a process model template.

Prerequisite	For more information, see ...
Define the trading partner profiles for the trading partners with whom you want to exchange business documents.	Chapter 4, “Defining Trading Partner Profiles in Trading Networks” in this guide and the <i>Building Your Trading Network</i> manual.
Import the PIP archives you need to the Integration Server.	Chapter 5, “Importing PIP Archives” in this guide.
Create your inbound and outbound mapping services.	Chapter 8, “Mapping a RosettaNet PIP Document” in this guide.
Create your Send to Back-End service.	
Create error and exception handlers, if necessary.	Chapter 11, “Handling Errors and Exceptions” in this guide. This chapter provides information about the types of errors that might occur.

Customizing a Process Model Template

To customize a process model template, you must first determine the appropriate process model template you need to use. The process model template that you use depends on two criteria:

- Your role in the PIP conversation (that is, initiator/sender or fulfiller/receiver).
- The number of transactions in the PIP conversation (that is, one-action or two-action).

For example, PIP 3A4 *Manage Purchase Order* would require the initiator or fulfiller version of the two-action asynchronous process model template. To determine whether a PIP is a one-action or two-action model, go to the RosettaNet Web site (<http://www.rosettanet.org>), and look up the specification for the PIP that you imported to the Integration Server.

After you determine the appropriate process model template to use, you are ready to customize the template.

 **To customize a process model template**

- 1 Start Modeler. If you need help with this step, see the *webMethods Modeler User's Guide*.



Note: To start Modeler, your Integration Server must be running.

- 2 Select **File** ► **Import**.
- 3 Go to:
`webMethods6\IntegrationServer\packages\WmRosettaNet\ProcessModels`
where `webMethods6\IntegrationServer` is the installation directory in which the Integration Server is installed.
- 4 Choose the process model template you want to work from, and click **Open**.
- 5 Select **File** ► **Save as New Version**.
- 6 Save the file under a different name than the template name.
- 7 For each step that waits for an action document, replace the RosettaNet Dummy Doc Type with the TN XML document type that you created for the step. For example, if you are using the `Two Action PIP Initiator.xml` process model template, you would replace the RosettaNet Dummy Doc Type for the **Wait for Internal Request Document** and **Wait for External Response Document** steps with the TN XML document types you created for these steps. When you replace a RosettaNet Dummy Doc Type with a TN XML document type, Modeler prompts you to assign the TN roles for the **Sender Role** and the **Receiver Role** to the TN XML document type. For information about TN roles, see [“TN Roles” on page 75](#) in this guide. For information about how to assign the TN XML document types that wait steps subscribe to, see the *webMethods Modeler User's Guide*.



Note: If you later change the name of an internal or external TN XML document type, you must also change the name of the TN XML document type in the process model to avoid an error.

- 8 If a wait step has a join type of “Complex,” replace the RosettaNet Dummy Doc Type for that join with the TN XML document type ID of the input business document assigned to that step.
 - a In the **Properties** dialog box, click the down arrow to the right of the **Complex** join type.
 - b In the **Join Type** list, click **Complex**.

- c In the **Complex Join Editor** dialog box, click **RosettaNet Dummy Doc Type**, and select the TN XML document type ID of the input business document assigned to the step.



Important! If you do not replace a RosettaNet Dummy Doc Type with the correct TN XML document type ID for a wait step with a join type of “Complex,” the incoming business document will not join the conversation.

- 9 Assign services; that is:
 - Assign outbound mapping and inbound mapping services to the **Outbound Mapping** and **Inbound Mapping** steps.
 - Assign error and exception handlers, if necessary,
 - Assign a service that will send a business document to your back-end system to the **Send to Back-End** step.

For information about how to assign the service that a step invokes, see the next section, [“Assigning the Service that a Step Invokes”](#).

- 10 Set timeout and retry count values if you want these values to be different than the PIP specification. For information about how to set timeout and retry values using Modeler, see [“Setting Timeout and Retry Count Values” on page 81](#) in this guide.
- 11 Enter a value for the **Focal Role** property. For information about focal roles, see [“Focal Role” on page 76](#) in this guide. For information about setting this property, see the *webMethods Modeler User’s Guide*.



Important! If a user’s focal role for a process is different than either of the TN roles for an incoming business document, the business document will not join the conversation.

- 12 Enter a value for the **Description** property. The description should briefly explain the use of the process model. For information about setting this property, see the *webMethods Modeler User’s Guide*.
- 13 Modify step labels and icons as appropriate. For information about how to modify step labels and icons, see the *webMethods Modeler User’s Guide*.
- 14 Save the process model.
- 15 Generate the process model; then update the process model for monitoring. For instructions about how to generate and update a process model, see the *webMethods Modeler User’s Guide*.

When you generate a process model, a process model package (containing services and triggers) and a process run-time script (or fragment) are created on the Integration Server.

- 16 In webMethods Monitor, enable the process model. For instructions about how to enable a process model, see the *webMethods Modeler User's Guide*.

Assigning the Service that a Step Invokes

When you customize a process model template, you need to assign the services that the **Inbound Mapping** and **Outbound Mapping** steps invoke and the service that the **Send to Back-End** step invokes. If you create your own error and exception handlers, you will also need to set these services as the ones that your error and exception steps invoke.

To assign the service that a step invokes

- 1 Start Modeler. If you need help with this step, see the *webMethods Modeler User's Guide*.
- 2 Open the process model template that you want to customize.



Note: Remember to save the process model template under a different name.

- 3 Right-click on the step for which you want to set the service to invoke.
- 4 Click **Select service to invoke**.
- 5 In the **Select service to invoke from server: Design Server** dialog box, browse the package folders, and select the service you want to invoke.
- 6 Save your changes.

For information about other process model properties that you can change in Modeler, see the *webMethods Modeler User's Guide*.

Setting Timeout and Retry Count Values

You can set both timeout and retry count values in Modeler. *Timeout* refers to the amount of time to wait to receive a Receipt Acknowledgment (ACK) or to receive a Response document, such as a POA. *Retry Count* refers to the number of times to attempt to send a Receipt ACK or to send a business document, such as a PO or POA, including the initial attempt to send the document.

The retry count value you set for a particular step determines the behavior of the process model at that step. The following table lists the possible retry count values and the effect of the value on a process model step.

Retry Count Value	Effect of Retry Count Value on the Process Model Step
0	This value provides the step with an unlimited number of retries.
1	This value provides the step with one try only. On the second retry, the process fails.
>1	For any value greater than 1, a “retry count exceeded” can take place. For example, if you set the retry count value to 2, and the process transitions to the step for a third attempt, the retry count is exceeded. For any value greater than 1, you must draw a “retry exceeded” transition from the step in question.

You can set the retry count value for each step independently of other steps. You should be concerned most with setting the retry count value for steps where a process model is sending an action message to a trading partner. Setting the value of the **Retry Count** property in Modeler changes the value for your PIP conversation but does not change the value in your trading partner’s PIP conversation.

To set timeout and retry count values

- 1 Start Modeler. If you need help with this step, see the *webMethods Modeler User’s Guide*.
- 2 Open the process model template that you want to customize.



Note: Remember to save the process model template under a different name.

- 3 Right-click on the step for which you want to set the timeout or retry count values.
- 4 Click **Properties**.
- 5 In the **Retry Count** field, enter the retry count.
- 6 In the **Timeout Value** field, enter the timeout (in milliseconds).
- 7 Save your changes.

For information about other process model properties that you can change in Modeler, see the *webMethods Modeler User’s Guide*.

Running and Monitoring a Conversation

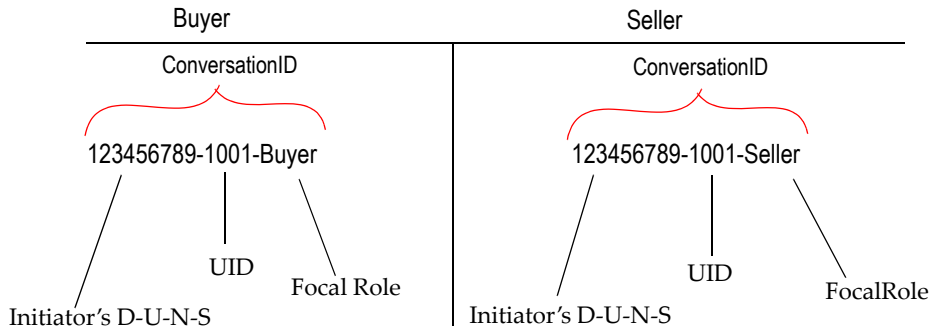
■ Running a Conversation and the ConversationID	84
■ Running a Conversation Example	85
■ Why Monitor a Conversation?	87
■ Sources of Conversation Status and Information	87
■ Archiving RNOs to the File System	89

Running a Conversation and the ConversationID

When a conversation starts, the process run time (PRT) selects the appropriate process model to use for the conversation. The PRT can run multiple instances of the same process model at one time. Each instance is a different conversation. To keep the instances of the conversation distinct, each instance must have its own unique ConversationID. webMethods assigns each PIP-transaction conversation a unique ConversationID.

This ConversationID is the concatenation of the initiator’s Global Business Identifier (a D-U-N-S® Number), a Process Instance Identifier, and the focal role of the process model, separated with hyphens. For example, if you are sending a Purchase Order (PO) Request to initiate a PIP 3A4 conversation, the webMethods RosettaNet Module creates a ConversationID for that conversation in the form of your D-U-N-S number, a Process Instance Identifier, and “Buyer” (because “Buyer” is the focal role of the initial process model for PIP 3A4). When the PORequest arrives at the fulfiller’s webMethods RosettaNet Module site, the webMethods RosettaNet Module creates a ConversationID for that conversation in the form of your D-U-N-S Number (because you were the initiator) the same UID (which is sent as part of the RosettaNet business message), and “Seller” (because “Seller” is the focal role of the process model for PIP3A4 on the fulfiller side when a PORequest is received. The following figure illustrates this concept.

ConversationID



For the PRT to associate documents in a single conversation with the correct running instance of a process model, all documents entering the conversation must specify the same ConversationID. In a process model, the Send to Back-End step (the step that sends the business document to the back-end) transitions to the back-end. When this occurs, the back-end system is passed information, including the ConversationID. For the conversation to continue properly when the back-end system returns the response, the business documents from the back-end system must include the correct ConversationID. This means that, at design time, when you set up your back-end systems to participate in a conversation, you must ensure that the back-end system maintains the ConversationID when it receives a business document.

You must maintain the ConversationID on your back-end system when you receive a business document. You then must include the ConversationID in any Response document you send back so that the webMethods RosettaNet Module can identify the proper conversation to which the Response document belongs and make sure the Response document rejoins the correct existing conversation.

XML Queries Used To Extract and Generate the ConversationID

When the webMethods RosettaNet Module receives a business document, it invokes the Trading Networks `wm.tn.doc:recognize` service. If the business document is from a back-end system, the `wm.ip.cm:process Document` service extracts the initiating trading partner's D-U-N-S Number, a process instance ID number, and the focal role of the transaction from the business document and uses these elements to generate the ConversationID, as explained in the previous section. The following sections illustrate the XML queries used to retrieve the elements that make up a ConversationID.

Initiator's D-U-N-S Number

The query to extract the initiator's D-U-N-S Number is:

```
/ServiceHeader[0]/ProcessControl[0]/ProcessIdentity[0]/initiatingPartner[0]/GlobalBusinessIdentifier[0]
```

Process Instance Identifier

The query to extract the Process Instance Identifier is:

```
/ServiceHeader[0]/ProcessControl[0]/ProcessIdentity[0]/InstanceIdentifier[0]
```

Focal Role

The query to extract the Focal Role is:

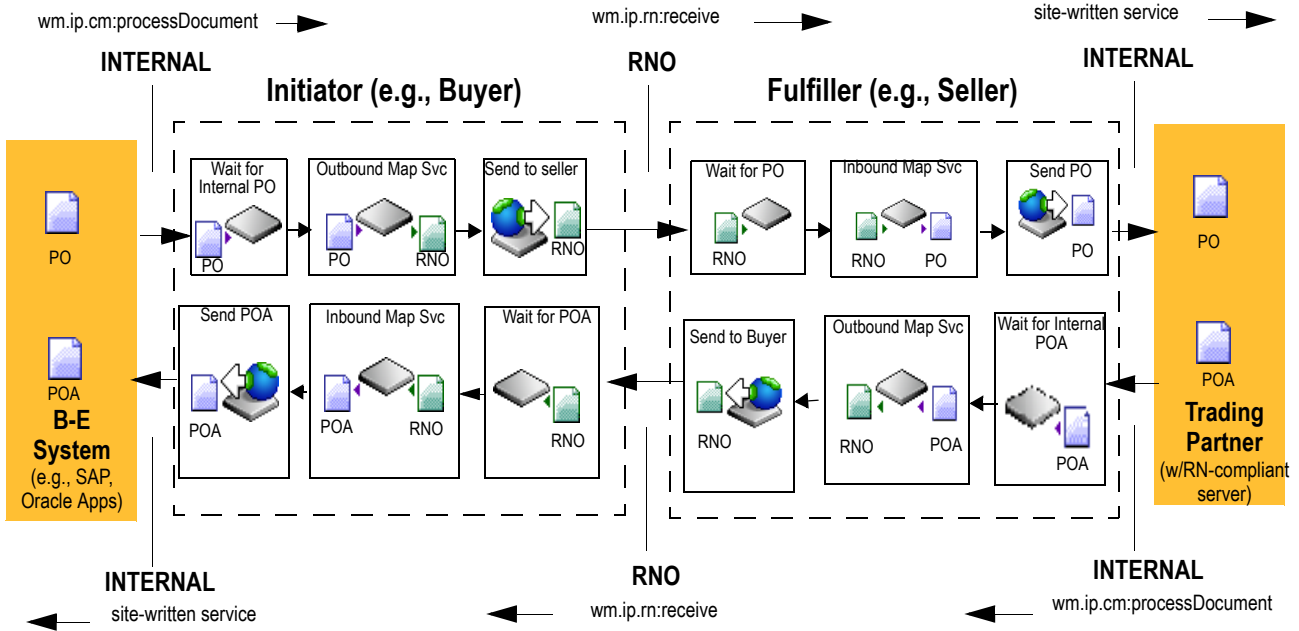
```
/ServiceHeader[0]/ProcessControl[0]/TransactionControl[0]/PartnerRoleRoute[0]/toRole[0]/PartnerRoleDescription[0]/GlobalPartnerRoleClassificationCode[0]
```

Running a Conversation Example

The following figure illustrates two distinct conversations: a conversation on the initiator's/buyer's side and a conversation on the fulfiller's/seller's side. The figure shows that an initiator and a fulfiller in a conversation both send RosettaNet PIP documents, or RosettaNet objects (RNOs), to each other by sending the document to `wm.ip.m:receive`. To send an internal business document from a back-end system to the webMethods RosettaNet Module, the initiator and the fulfiller use `wm.ip.cm:processDocument`. To send a business document from the webMethods RosettaNet Module to a back-end system, the

initiator and the fulfiller use a site-written service. The figure uses PIP3A4 as an example. For further explanation, see the sections that follow the figure.

Running a PIP 3A4 Conversation



Starting a Conversation

To start a conversation on the initiator’s side, the initiator’s back-end system calls `wm.ip.cm:processDocument`. To start a conversation on the fulfiller’s side, a service invoked by the initiator’s **Send to Seller** step sends the PO (in RNO format) to `wm.ip.rn:receive`. Both `wm.ip.cm:processDocument` and `wm.ip.rn:receive` invoke `wm.tn.doc:recognize` to recognize the TN XML document type of the document received. On each side, a `BizDocEnvelope` is created. Both `wm.ip.cm:processDocument` and `wm.ip.rn:receive` send the `BizDocEnvelope` to the PRT.

On each side, the PRT checks to see if a `ConversationID` exists for the conversation. If no `ConversationID` exists, the webMethods RosettaNet Module creates a `ConversationID`. Then the PRT creates a new instance of the process model and assigns to the process model the `ConversationID`. The PRT executes the conversation based on the steps defined in the process model.

Rejoining an Existing Conversation on the Fulfiller’s Side

When the process model on the fulfiller’s side has completed processing a business document, a service invoked by the fulfiller’s **Send PO** step sends the business document to

a back-end system. The back-end system must maintain the ConversationID associated with the PO. The back-end system processes the PO and sends an internal Purchase Order Acceptance (POA) to rejoin the existing conversation by invoking `wm.ip.cm:processDocument`. You pass the internal POA and the stored ConversationID as inputs to this service. The PRT searches for a running conversation (or process instance) where a **Wait for POA** step is waiting for a POA with a matching ConversationID. When the PRT finds the correct conversation (or process), it sends the POA to rejoin the existing conversation.

Rejoining an Existing Conversation on the Initiator's Side

To send the POA to the initiator, the fulfiller sends the document (in RNO format) to `wm.ip.m:receive`. The `wm.ip.m:receive` service invokes `wm.ip.tn:recognize`, which recognizes the document and creates a BizDocEnvelope. The `wm.ip.m:receive` service sends the BizDocEnvelope to the PRT. The PRT searches for a running conversation (or process instance) where a **Wait for POA** step is waiting for a POA with a matching ConversationID. When the PRT finds the correct conversation (or process), it sends the POA to rejoin the existing conversation.

If the PRT does not find a running conversation where a **Wait for POA** step is waiting for a POA with a matching ConversationID, the PRT ignores the POA.

Why Monitor a Conversation?

You monitor a RosettaNet conversation to track the state of a particular RosettaNet transaction. For example, suppose that you initiate a PIP 3A4 *Manage Purchase Order* transaction by sending a PO for 30 PCs to a trading partner. The next day, the trading partner calls, indicating that they did not receive your PO. In the webMethods RosettaNet Module, you can view the activity of the conversation to determine what the current status is and what activity has occurred during the progress of the conversation. Following the same example, a reason why the trading partner did not receive the PO could be because the transaction encountered an error, such as a timeout error. By monitoring the conversation, you can determine if this was the cause for non-receipt of the PO. You can see whether your PO Request was resent, and how many times, and with what success. By viewing the current status of and the progress of a PIP transaction, you can take appropriate action, which might include retrying the conversation, editing your trading partner profiles, or editing your process models.

Sources of Conversation Status and Information

To monitor conversations, you have a number of sources of information from which you can draw to determine the status of your conversations and the activity of the various involved software entities. The following table lists and describes the primary sources from which you can draw information.

Source	Description
webMethods Monitor	Use this tool to monitor and manage conversations. For information about Monitor, see the next section, “About Monitor” , and the <i>webMethods Monitor User’s Guide</i> .
webMethods Trading Networks Transaction Analysis screen	Use this log to query and analyze results of documents that are sent or received by Trading Networks. For information about transaction analysis, see the <i>Building Your Trading Network</i> manual.
webMethods Integration Server error log	Use this log to retrieve server-related error and exception messages that occur during an invocation of a service directly or indirectly from a conversation.

About Monitor

Monitor is a web-based user interface that you can use to examine instances of your process models. Monitor displays information about instances of your process models by accessing data from the Process Logging Database. For information about the Process Logging Database, see [“webMethods RosettaNet Module Run-Time Architecture/Components” on page 20](#) in this guide.

Using Monitor, you can:

- Search for a conversation by name, status, or date range.
- Search specifically for a conversation that ended in error.
- See a graphical overview of a conversation.
- Examine information about a conversation and its execution; for example, status of the conversation, status and iteration of process model steps, and so on.
- View services used in the conversation.
- Perform control tasks to affect the state of a conversation:
 - Suspend and resume a conversation.
 - Enable and disable a conversation.
 - Start and stop a conversation.
 - Edit and resubmit process model steps.

For more information about Monitor and for procedures about how to perform the aforementioned tasks, see the *webMethods Monitor User’s Guide*.

Archiving RNOs to the File System

As a default, the webMethods RosettaNet Module archives all business documents that it sends and receives to the Trading Networks database. In addition, you can configure the webMethods RosettaNet Module to archive all business documents it receives to the file system. For example, you can configure each transport protocol, RNIF 1.1 and RNIF 2.0, to archive either inbound RNOs, outbound RNOs, or both. You can configure each transport protocol to store the archived RNOs where you specify. You can retrieve these archived RNOs by going to the archive directory and opening the archived RNOs in an editor of your choice.

Archiving RNOs to the file system is an added level of archiving provided for debugging purposes. If performance is a concern, however, you should turn this level of archiving off.

To archive inbound or outbound RNOs to the file system

- 1 Open either or both of the following files in the text editor of your choice:

`webMethods6\IntegrationServer\packages\WmRNIF11TRP\config\rnif11.cnf`

`webMethods6\IntegrationServer\packages\WmRNIF20TRP\config\transport.cnf`

where `webMethods6\IntegrationServer` is the directory in which your Integration Server is installed.

- 2 Change the value of the *archiveInboundRNO?* Parameter to **yes**.

Note that the default value is already set to `yes` so that the RNOs will be archived.

- 3 Change the value of the *InboundFolder* parameter to the directory you want to contain your archived inbound RNOs.

Note that the default value is already set to `packages/WmRNIF11TRP/pub/archive` for the RNIF 1.1 transport protocol, or to the corresponding directory for the RNIF 2.0 transport protocol.

- 4 Change the value of the *archiveOutboundRNO?* Parameter to **yes**.

Note that the default value is already set to `yes` so that the RNOs will be archived.

- 5 Change the value of the *OutboundFolder* parameter to the directory you want to contain your archived outbound RNOs.

Note that the default value is already set to `packages/WmRNIF11TRP/pub/archive` for the RNIF 1.1 transport protocol, or to the corresponding directory for the RNIF 2.0 transport protocol.




Note: RNO archive file names are composed of a series of parts, separated by an underscore. The first part of the name is the D-U-N-S Number of the trading partner who sent the RNO to you or received the RNO from you. The second part of the name is the Process Instance ID of the conversation to which the RNO belonged. The third part of the name is either an *S* or an *R* to indicate whether the business document has been sent or received, respectively. The fourth part of the name is the PIP number and version that was implemented by the conversation associated with the RNO. The fifth part of the name is the RosettaNet Global Business Signal Code for signal documents (for example, Receipt ACK, Exception, and so on) and Global Business Action Code for business documents (for example, PIP 3A4 Purchase Order Request). The last part of the name is the file extension of *.rno*.

For example, an RNO might have a name like the following:

```
987654321_0a013218f70dc0a6000000e5_R_3A4v1.1_ReceiptAcknowledgment.rno
```

- 6 Save and close the configuration file.

The webMethods RosettaNet Module will now archive your RNOs according to the settings you specified in the configuration files. If at any time you want to turn off the archival feature, use this same procedure to set the *archiveInboundRNO?* and *archiveOutboundRNO?* values to **no**.

- 7 Ensure the Integration Server and Server Administrator are running.
- 8 In the Server Administrator navigation panel, under the **Packages** section, click **Management**.
- 9 For the *WmRNIF11TRP* and *WmRNIF20TRP* packages, click **Reload**  .

Handling Errors and Exceptions

■ Handling Errors and Exceptions	92
■ When Might Errors or Exceptions Occur?	94
■ Initiating an Error Process Manually	97
■ Where to Find Information About Errors	98

Handling Errors and Exceptions

The webMethods RosettaNet Module provides handler services for various tasks; for example, handling errors during inbound validation of a RosettaNet PIP document or handling a RosettaNet PIP document that does not conform to any TN XML document type. Optionally, you can customize the provided handler services or create your own handler services to enhance the messages written to the various logs, generate an e-mail message upon an exception, write messages to a database, and so on.

What Is a Handler Service?

A handler service is a service that a particular step invokes when an event in the PIP process occurs. For example, in the Two Action PIP Initiator Model.xml process model template, the **Verify ACK** step invokes the `wm.ip.cm.handlers.verifyAcknowledgement` service when the initiator receives a Receipt Acknowledgement (ACK) from the fulfiller.

In webMethods Developer, you create handler services by creating a new service for each event that you want to handle in a RosettaNet conversation.

Handler Services Provided With the webMethods RosettaNet Module

The following table lists and describes the handler services that the webMethods RosettaNet Module provides in the `WmIPRoot` package and identifies whether you can edit a particular handler service. For more information about any of these services, see the *webMethods RosettaNet Module Built-In Services Reference Guide*. You can find other handler services provided with the RosettaNet samples as well as other built-in handler services provided in the `WmRoot` and `WmPublic` packages.

Service	Description	Can Edit	Do Not Edit
<code>wm.ip.cm.handlers.defaultHandler</code>	The handler for generic situations. Causes a message to be written to the log.	✓	
<code>wm.ip.cm.handlers.end</code>	The last step in a conversation.	✓	
<code>wm.ip.cm.handlers.error</code>	The service called when an error occurs during the processing of a RosettaNet conversation.		✓
<code>wm.ip.cm.handlers.inboundValidation</code>	The service for validating inbound RosettaNet business documents.		✓

Service	Description	Can Edit	Do Not Edit
wm.ip.cm.handlers:processNOF	The service for processing a received Notification of Failure (NOF) and setting the related conversation to an error state.		✓
wm.ip.cm.handlers:processUnhandledDocument	The service for handling a RosettaNet PIP document that does not conform to any TN XML document type.		✓
wm.ip.cm.handlers:send	The service for sending RosettaNet PIP documents to a trading partner via the appropriate transport protocol.		✓
wm.ip.cm.handlers:sendReceiptACK	The service for sending the Receipt ACK upon receipt and validation of a RosettaNet PIP document.		✓
wm.ip.cm.handlers:sendSyncResponse	The service for invoking the appropriate sendSyncResponse based on the transport protocol being used by the conversation. For example, if the conversation uses the RNIF version 1.1 transport protocol, the service invokes the wm.ip.rnif11.trp:sendSyncResponse service that is provided in the WmRNIF11TRP package.		✓
wm.ip.cm.handlers:start	A placeholder for the step in a conversation.	✓	

Service	Description	Can Edit	Do Not Edit
wm.ip.cm.handlers:startNOF	The service for initiating a <i>Pip0A1NotificationofFailure</i> . This service invokes the appropriate startNOF service based on the transport protocol being used by the conversation. For example, if the conversation uses the RNIF version 1.1 transport protocol, this service invokes the <i>wm.ip.rnif11.trp:startNOFv1</i> service that is provided in the <i>WmRNIF11TRP</i> package.		✓
wm.ip.cm.handlers:verifyAcknowledgement	The service for invoking the appropriate <i>verifyAcknowledgement</i> service based on the transport protocol being used by the conversation. For example, if the conversation uses the RNIF version 1.1 transport protocol, the service invokes the <i>wm.ip.rnif11.trp:verifyAcknowledgement</i> service that is provided in the <i>WmRNIF11TRP</i> package.		✓



Note: If you **must** edit a handler service that webMethods has recommended you not to edit, you should be an advanced, experienced developer or get an advanced, experienced developer to edit the handler service.

When Might Errors or Exceptions Occur?

Errors or exceptions might occur at any point in a RosettaNet PIP interchange. They might occur when:

- RosettaNet PIP documents are created or sent.
- Signature verification fails.
- Encryption or decryption fails.
- RosettaNet PIP document components are packed into RosettaNet objects (RNOs) for transmission.
- RosettaNet PIP documents are unpacked into their components (headers, content, signatures, etc.).

- RosettaNet PIP documents are validated for structure and content.
- A RosettaNet transaction expires after its retries are exceeded.
- A RosettaNet PIP document is received out of sequence, violating the document order specified in the particular PIP.

General Errors With RosettaNet PIP Documents

The following table lists the general errors that might occur with a RosettaNet PIP document based on the RNIF (1.1 or 2.0) used and whether the business document is inbound or outbound.

RNIF Version	Inbound/Outbound	Error	How webMethods RosettaNet Module Responds to the Error
1.1 & 2.0	Inbound	Header validation fails.	The process run time (PRT) ignores the RosettaNet PIP document. Does not send an exception.
1.1 & 2.0	Inbound	Signature verification fails.	The PRT ignores the RosettaNet PIP document. Does not send an exception.
1.1 & 2.0	Inbound	Content validation fails.	The PRT places the conversation in an error state. Sends an exception to the trading partner with the reason for the exception in the text of the message.
1.1 & 2.0	Outbound	Validation fails.	The PRT ignores the RosettaNet PIP document. Does not send an exception. Sets the conversation to an error state. (This error can occur only if outbound validation is turned on.)
1.1 & 2.0	Outbound	Cannot package the RNO.	The PRT ignores the RosettaNet PIP document. Does not send an exception. Sets the conversation to an error state.
1.1 & 2.0	Outbound	Error occurs during HTTP/HTTPS transmission.	The webMethods RosettaNet Module retries the transmission the designated number of times. If the transmission fails on all attempts, the PRT places the conversation in an error state.
2.0	Inbound	Decryption fails.	The PRT ignores the RosettaNet PIP document. Does not send an exception.

RNIF Version	Inbound/ Outbound	Error	How webMethods RosettaNet Module Responds to the Error
2.0	Outbound	Any error that occurs when the transport protocol is in Test mode.	The PRT places the conversation in an error state and sends an exception to both trading partners (that is, the initiator and the fulfiller) so that both ends of the conversation see the error that is occurring as part of the debugging process. Note: The RNIF 2.0 transport protocol provides the Debug headers in the Test mode only.
2.0	Inbound	Any error that occurs when unpacking an RNO and while the transport protocol is in Test mode.	The PRT places the conversation in an error state and sends an exception that describes the error to the trading partner who sent the RosettaNet PIP document.

Errors When a Transaction Times Out: Notification of Failure

When a transaction you are conducting with your trading partner expires, or times out, you should initiate PIP 0A1 *Notification of Failure*. PIP 0A1 is designed to communicate between trading partners when such errors occur. The PIP communicates the errors that occur at the various stages of processing business documents so that both trading partners know the problems that occurred and where to look to fix the problems.

The following is an example of a failed conversation: After your enterprise receives the Receipt Acknowledgement (ACK) message following your initial Request document during a conversation, the conversation waits for a Response document from your trading partner. If, after 24 hours, you have not received the Response document from your trading partner, the conversation in your enterprise expires. Your enterprise should initiate PIP 0A1 to notify your trading partner that you have not received a Response document and that you believe a problem exists on their side. For information about setting timeout values for a conversation, see [“Setting Timeout and Retry Count Values” on page 81](#) in this guide.

The webMethods RosettaNet Module sends an NOF to the alternate delivery method that you specified for the *Alternate Protocol* parameter in the Trading Partner Agreement (TPA) for PIP 0A1 *Notification of Failure*. This alternate delivery method is used because the problem that you are having communicating with a trading partner might be related to the URL from the primary delivery method that you are using (that is, your trading partner’s Web server has failed or the URL in your trading partner’s profile has become corrupted, or an ISP link between the two of you has failed, and so on). If you did not specify an alternate delivery method for the *Alternate Protocol* parameter, the webMethods RosettaNet Module uses the preferred delivery method that you selected in the profile for the trading partner. If you did not specify a preferred delivery method, an error occurs.

To send an NOF to your trading partner, you must set up the process model for your other PIPs so that they initiate PIP 0A1 for the appropriate error conditions in those other PIPs.

All of the PIPs that webMethods provides are already configured to invoke PIP 0A1 when they encounter the appropriate error conditions. In addition, your trading partners must configure their webMethods RosettaNet Module installations to accept NOFs.

Errors When a Business Document Is Out of Sequence

A process model specifies steps exactly as they are to happen based on the results of the execution of each step. Some steps invoke services to create outbound business documents or to process inbound business documents. Other steps wait for business documents to be received from trading partners.

If you receive a business document from a trading partner, but either you did not expect a business document (that is, the conversation is not waiting for a business document) or the business document that you received is not the business document that you expected (that is, you receive the wrong business document), the PRT handles the received business document as an out-of-sequence or unhandled business document. The arrival of an unhandled business document causes the PRT to place the conversation in an error state by sending the conversation to the **Error** step. The **Error** step invokes the `wm.ip.cm.handlers:processUnhandledDocument` service in the `WmIPRoot` package. This service generates a general exception and sends the exception to the trading partner who sent the unexpected business document.

An example might be when the process model specifies the business documents to send and receive in a certain order, but the order in which the conversation actually sends and receives the business documents becomes scrambled during the communications over the Internet. This could cause a business document to arrive before it is scheduled, according to the specified order.

For details on error handling, see [“Handler Services Provided With the webMethods RosettaNet Module” on page 92](#) in this guide and the *webMethods RosettaNet Module Built-In Services Reference Guide*.

Initiating an Error Process Manually

You might want to manually initiate an error process to:

- **Void an already completed transaction.** For example, suppose that you, as the initiator, successfully execute a RosettaNet PIP 3A4 transaction with one of your trading partners. The conversation ends when your trading partner receives the final Receipt ACK successfully from you, indicating that you have received their Purchase Order Acceptance (POA). However, after the conversation ends, you encounter a semantic error in their POA (for example, the POA states that you ordered four PCs instead of only two, as you stated in your PO Request).

To notify the trading partner that you intend to void the entire transaction, you can manually initiate PIP 0A1 *Notification of Failure* as well as set the transaction to an

error state by invoking `wm.ip.rnif.util.privateProcessFailureInitiateNof` for RNIF 1.1 or `wm.ip.rnif20.util.privateProcessFailureInitiateNof` for RNIF 2.0.

- **Cancel a transaction in progress.** For example, suppose that you, as the seller, have received a PIP 3A4 PO Request from your trading partner and have sent a Receipt ACK. While processing the PO in your back-end, you realize that you have a semantic problem with the PO received. To notify your trading partner of the error, you can manually send a General Exception to your trading partner as well as set the current conversation to an error state by invoking `wm.ip.rnif11.util.privateProcessExceptionRNIF11` for RNIF 1.1 or `wm.ip.rnif20.util.privateProcessExceptionRNIF20` for RNIF 2.0.

Where to Find Information About Errors

You can view information about transactions and conversations that are in various error states by looking at the webMethods Trading Networks **Transaction Analysis** screen and webMethods Monitor. For more information about these resources, see [“Sources of Conversation Status and Information” on page 87](#) in this guide.

For sample handler services for Receipt ACKs and exceptions, see the `WmRNSample` package.

Processing Large Business Documents

■ Limits to Processing Large Business Documents	100
■ Customizing to Enable Processing of Large Business Documents	101
■ Now That You Have Made Your Customizations, What Happens During Run Time?	108
■ Large Business Document Sample	111

Limits to Processing Large Business Documents

The default document processing facilities provided with the webMethods RosettaNet Module readily handle RosettaNet PIP documents up to a size of 15 MB or so on an average (1-1.5 GB memory) installation. The default facilities for processing larger documents with the webMethods RosettaNet Module, however, are currently limited by the method by which the webMethods Integration Server creates IData objects and by the fact that the pipeline persists all documents, regardless of their size, to the Integration Server Repository.

The IData Objects Limitation

The webMethods Integration Server uses IData objects to represent XML documents so that they can be more easily processed than native XML documents. However, the overhead associated with creating an IData object in the webMethods Integration Server becomes substantially greater as the XML document becomes larger.

Most large XML documents, however, are composed of some small amounts of non-repeated information and a large number of repeated sections of XML. For example, a PIP 3D8 document contains repeated sections of the *workInProgress* line item.

The Integration Server provides a set of services, such as the `pub.web:getNodeIterator` service, to aid in processing large, repetitive XML documents. These services allow you to convert sections, particularly the repeating sections, of large XML documents into IData objects. This way, for example, the Integration Server can process each *workInProgress* line item of a PIP 3D8 document, one at a time, rather than all at once. For more information about the `pub.web:getNodeIterator` service, see the *webMethods Built-In Services Reference Guide*.

The Pipeline Limitation

The pipeline is persisted at every wait step. If a large business document exists in the pipeline, the entire business document is persisted to the Integration Server Repository, possibly multiple times. Saving such large business documents/messages to the Integration Server Repository and then restoring them back from the Repository consumes a lot of memory.

Customizing to Enable Processing of Large Business Documents

To enable the webMethods RosettaNet Module to process large XML documents of up to 100 MB, during design time, you customize the items in the following table based on whether you are the initiator or the fulfiller in the conversation and whether the PIP transaction is a one-action or two-action PIP transaction:

Customize this ...	One-Action PIP Transaction		Two-Action PIP Transaction	
	Initiator	Fulfiller	Initiator	Fulfiller
PIP	✓	✓	✓	✓
Outbound Mapping Service	✓		✓	✓
Inbound Mapping Service	✓	✓	✓	✓
Validation Service		✓	✓	✓

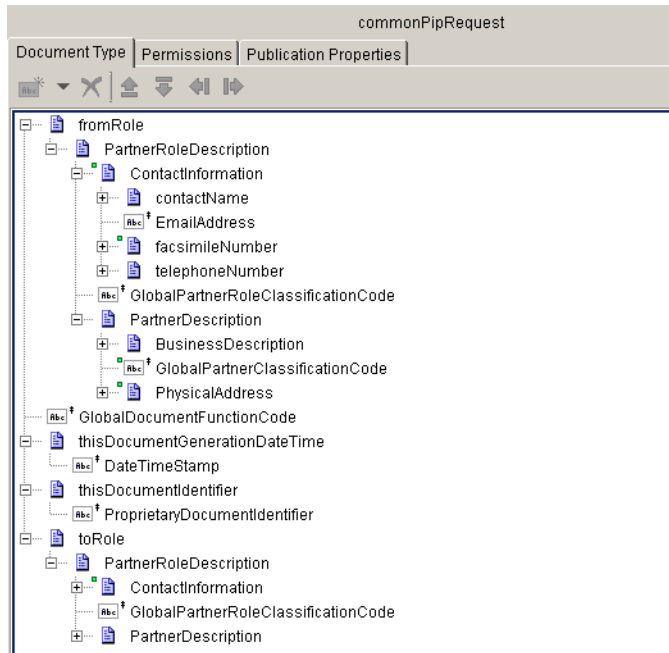
The following sections instruct you how to customize the PIPs and mapping and validation services to enable the webMethods RosettaNet Module to process large business documents. The sections include examples from the PIP 3D8 Large Document Sample. For information about mapping services, see [Chapter 8, “Mapping a RosettaNet PIP Document”](#), in this guide.

Step 1: Separate the RosettaNet PIP Document into Non-Repeating and Repeating Parts

To process a large RosettaNet PIP document, you must separate its elements into non-repeating and repeating parts. This separation is necessary to be able to validate the PIP in multiple chunks. For example, in the WmRNLargeDocSample package, PIP 3D8 is split into a non-repeating and a repeating part.

The non-repeating part, commonPipRequest, is illustrated in the following figure. For further explanation, see the text that follows the figure.

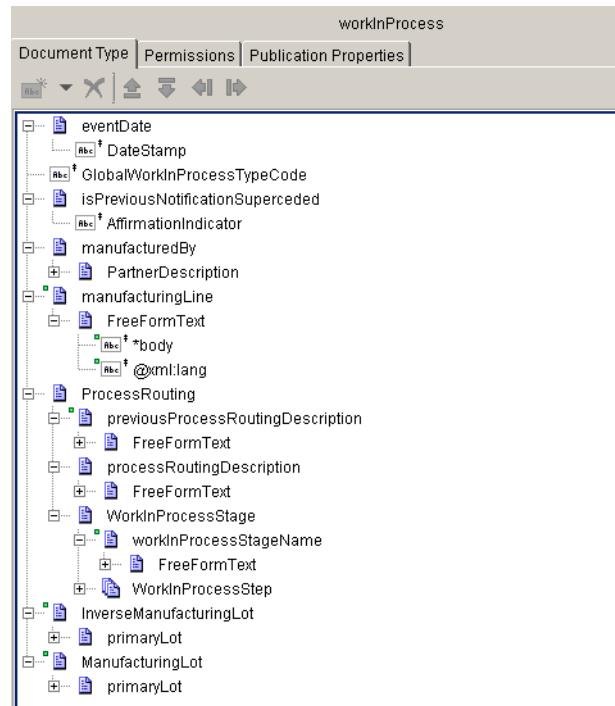
commonPipRequest Non-Repeating Section



commonPipRequest contains information that you would list only once in a PIP 3D8 document; for example, sender and receiver contact information, descriptions, and Global Partner Classification Codes.

The repeating part of PIP 3D8, `workInProgress`, is illustrated in the following figure. For further explanation, see the text that follows the figure.

workInProgress Repeating Section



`workInProgress` contains types of information that you might list repeatedly in a PIP 3D8 document for perhaps hundreds of product components; for example, process routing, manufacturing line, and work-in-process stage of a particular component.

Step 2: Customize the Outbound Mapping Service

You can either customize an existing mapping service or create a new mapping service to handle large business documents. Perform the following procedure to customize an existing outbound mapping service.

To customize the outbound mapping service

- 1 Invoke `wm.ip.tspace:createFile` to open a reservation in *TSpace* (temporary hard drive disk space). This service returns a reservation object that points to a file on the file system. For information about `wm.ip.tspace:createFile` see the *webMethods RosettaNet Module Built-In Services Reference Guide*.
- 2 Reading from the back-end, map the non-repeating parts of the PIP into an RNO in *TSpace*.

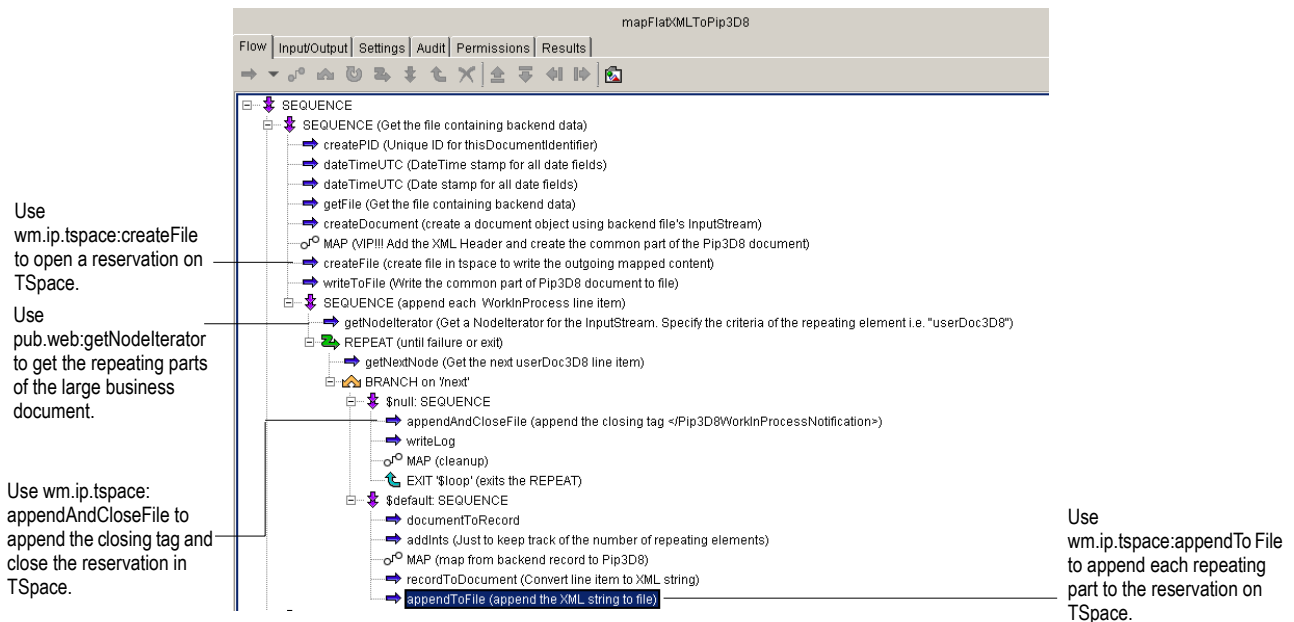


Note: It is recommended that you write the PIP to *TSpace* as the PIP is laid out. For example, the non-repeating parts might be at the end of a PIP. In that case, you would first write the repeating parts to *TSpace* and then write the non-repeating parts to *TSpace*.

- 3 Reading from the back-end, map the repeating parts of the PIP into an RNO in *TSpace* by invoking `pub.web:getNodeIterator`, a public built-in service that you use to loop over an array of `IData` objects. For information about the `pub.web:getNodeIterator` service, see the *webMethods Built-In Services Reference Guide*.
- 4 Write the end tag that corresponds to the root tag of the business document.
- 5 Close the reservation in *TSpace*.
- 6 Invoke `wm.tn.doc:addContentPart` to prevent the reservation object you created from being garbage collected and to persist it to the `BizDocEnvelope`, which is already saved in the `webMethods Trading Networks` database.
- 7 Map the reservation object to the `documents\Payload IData` object.

The following figure illustrates an outbound mapping service from the PIP 3D8 Sample that handles large business documents.

Outbound Mapping Service that Handles Large Business Documents

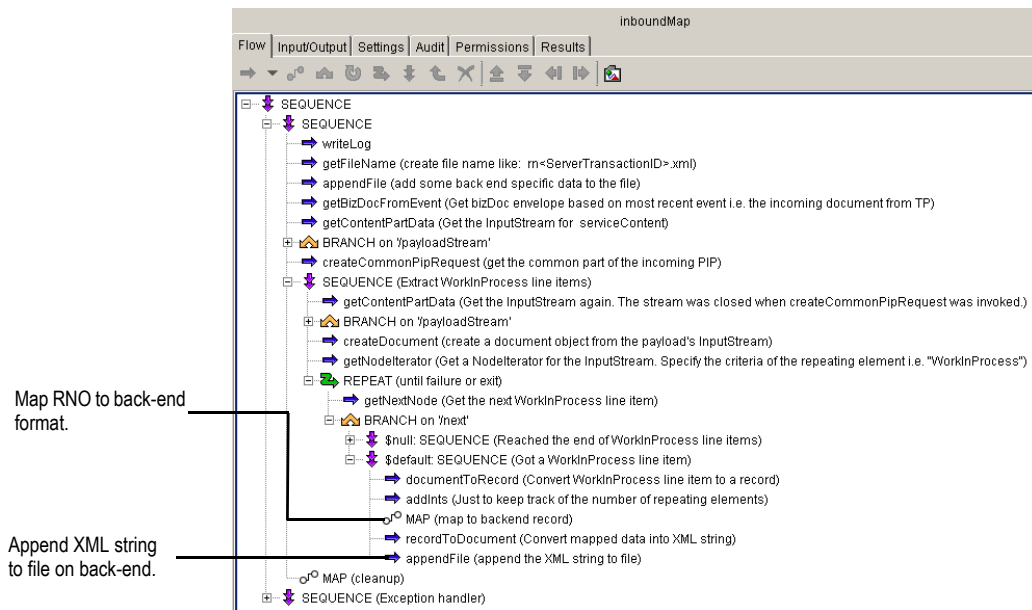


Step 3: Customize the Inbound Mapping Service

The inbound mapping service performs the reverse action of the outbound mapping service. In your inbound mapping service, you map an RNO to a back-end format.

The following figure illustrates an inbound mapping service from the PIP 3D8 Sample that handles large business documents. For further explanation, see the text that follows the figure.

Inbound Mapping Service that Handles Large Business Documents



The mapping service retrieves the payload from the BizDocEnvelope and then retrieves the non-repeating section. Then, it creates a document object and a node iterator from the payload stream. Next, it loops through each node and maps the node to the back-end object.

Step 4: Customize the Validation Service

You can either customize an existing validation service or create a new validation service to handle large business documents. Perform the following procedure to customize an existing validation service.



Note: If your process model handles both large and small business documents, then your validation service must handle both large and small business documents.



To customize the validation service

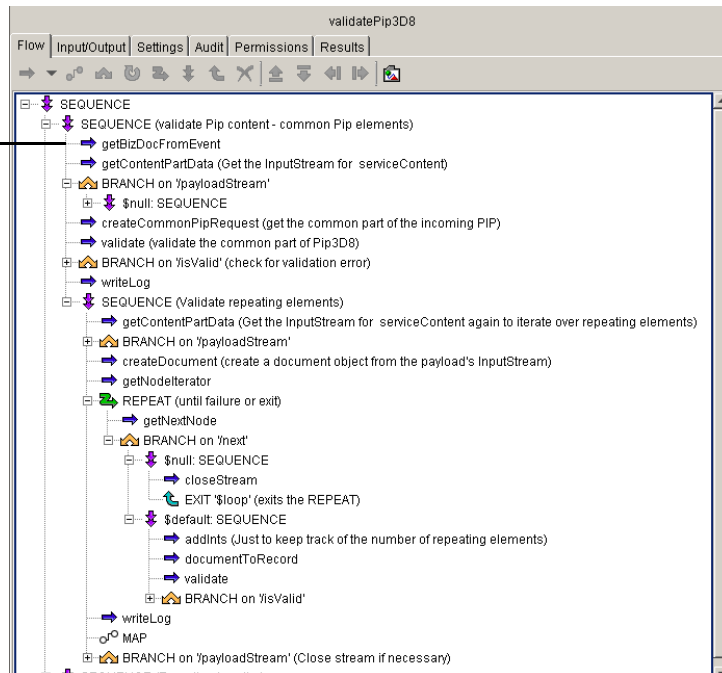
- 1 Retrieve the BizDocEnvelope from the Trading Networks database.
- 2 For the non-repeating parts, convert the XML strings to IData objects, and validate.
- 3 For the repeating parts, convert the XML strings to IData objects and validate.

If either step “2” or step “3” returns a value of *false*, then the validation service has failed.

The following figure illustrates the validation service from the PIP 3D8 Sample that handles large business documents.

Validation Service that Handles Large Business Documents

Use `wm.ip.util.getBizDocFromEvent` to retrieve a BizDocEnvelope from the Trading Networks database



- In the Trading Partner Agreement (TPA) for the PIP, enter this validation service name as the value for the **ValidateService** parameter.



Note: The webMethods RosettaNet Module accesses the validation service during the inbound validation step for an inbound business document. For an outbound business document, the webMethods RosettaNet Module accesses the validation service if the value for the **ValidateOutput** parameter is *yes*.

The following figure illustrates the **Input for 'wm.ip.rn.rec:UserParameters'** screen in the Trading Networks Console, where you enter the validation service that you just created or customized to handle large business documents.

Input for 'wm.ip.rn.rec:UserParameters' Screen in the Trading Networks Console

Enter the name of the validation service that handles large business documents here.

Now That You Have Made Your Customizations, What Happens During Run Time?

During run time, you send a business document from the back-end system to the webMethods RosettaNet Module. The webMethods RosettaNet Module processes the business document. That is, it invokes a webMethods Trading Networks service to recognize the TN XML document type and create the BizDocEnvelope and to send the BizDocEnvelope to the process run time (PRT).

In the aforementioned scenario, you could send either a small business document or the large business document from the back-end to the webMethods RosettaNet Module. For example, in the PIP 3D8 Large Document Sample, a small document is sent from the back-end system that includes a file name that references the large business document. Or you could send the large business document, and after Trading Networks creates the BizDocEnvelope, open a reservation to *TSpace* and use `wm.tn.doc.getContentPart` to retrieve the content of the large business document and write it to *TSpace*. See the *WmTNSample* package for this type of handling of large business documents.

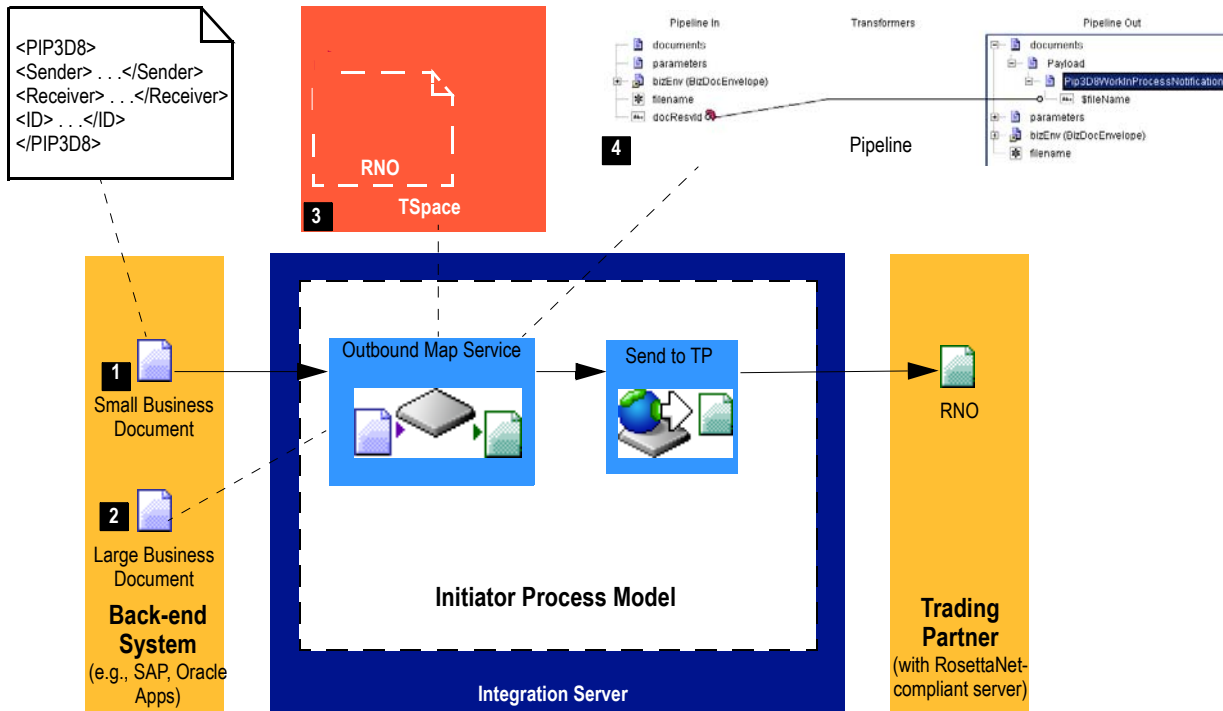
However, using a small business document to initiate a RosettaNet transaction for a large business document is the better of the two methods. By sending a small business document, you can improve performance because you do not need to load the entire large business document into memory.

The following two sections explain the outbound mapping and validation of the large business document in the PIP 3D8 Large Document Sample.

Outbound Mapping of a Large Business Document

The following figure illustrates the outbound mapping of a large business document. For further explanation, see the table that follows the figure.

Outbound Mapping of a Large Business Document



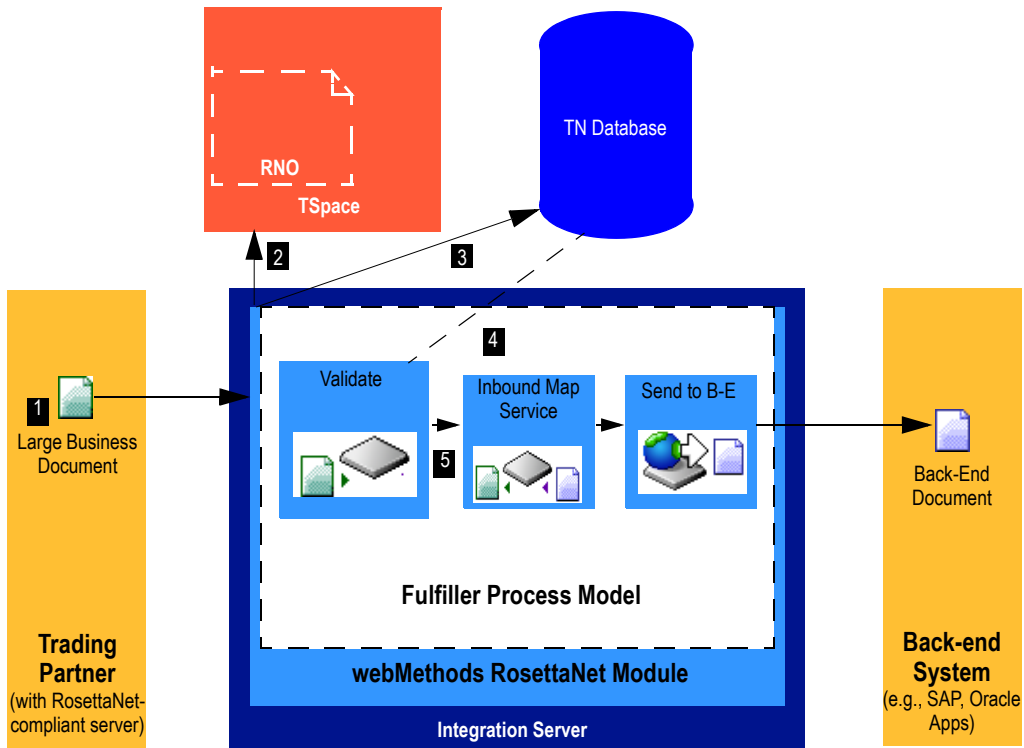
Step	Description
1	A back-end system sends an outbound small business document that references a large business document to the webMethods RosettaNet Module. In the Initiator Process Model, the small business document is passed to an outbound mapping step.
2	An outbound mapping service uses the information provided in the small business document to read the non-repeating and repeating parts of the large business document from the back-end system through a Java InputStream.

Step	Description
3	As the outbound mapping service reads each part of the large business document from the back-end system, it writes the RNO to <i>TSpace</i> . In this way, the large business document gets mapped to an RNO, which is stored in <i>TSpace</i> , instead of using up valuable memory. After writing the RNO to <i>TSpace</i> , the outbound mapping service closes the file.
4	The outbound mapping service then maps the filename presented by <i>docResvId</i> into the outgoing business document with the variable name of <i>\$fileName</i> as illustrated in the previous figure.

Inbound Validation of a Large Business Document

The following figure illustrates the inbound validation of a large business document. For further explanation, see the table that follows the figure.

Inbound Validation of a Large Business Document



Step	Action
1	A trading partner sends a large business document to the webMethods RosettaNet Module at a fulfiller site. The webMethods RosettaNet Module recognizes that the business document is a large business document by the value in the <i>tn.BigDocThreshold</i> property in the webMethods Trading Networks properties.cnf file. For more information about this file, see the <i>webMethods Trading Networks Large Document Handling</i> paper.
2	The webMethods RosettaNet Module stores the large business document in <i>TSpace</i> .
3	The webMethods RosettaNet Module saves the large business document in the Trading Networks database.
4	During the conversation, a validation service at the Validation step reads the non-repeating part and repeating parts of the large business document by a Java input stream from the Trading Networks database. The validation service validates the non-repeating part and the repeating parts.
5	After validation, the large business document is passed to the Inbound Map Service step, where processing continues.

Large Business Document Sample

The webMethods RosettaNet Module includes the `WmRNLargeDocSample` package, which shows how to map inbound and outbound large business documents, how to validate large business documents, and how to persist and retrieve large business documents from the Trading Networks database. The `WmRNLargeDocSample` package uses PIP 3D8 *Distribute Work in Process* to demonstrate these tasks.

If you have not already done so, install the `WmRNLargeDocSample` package using the standard Integration Server package installation procedure. For instructions about installing a package on the Integration Server, see the *webMethods Integration Server Administrator's Guide*.

Running the PIP 3D8 Sample

The PIP 3D8 Sample demonstrates sending a 500KB PIP 3D8 document as an example of a large business document.

To run the PIP 3D8 Sample

- 1 In the properties.cnf file, set *tn.BigDocThreshold* = "500000" on both the initiator/sender and the fulfiller/receiver sides. The properties.cnf file is located at:

```
webMethods6\IntegrationServer\packages\WmTN\config
```

where `webMethods6\IntegrationServer` is the directory in which the Integration Server is installed. If you do not see the `tn.BigDocThreshold` property in the `properties.cnf` file, add it. For information about the `tn.BigDocThreshold` parameter, see the *webMethods Trading Networks Large Document Handling* paper.

- 2 Copy the `SamplePip3D8.par` file from the `WmRN\LargeDocSample\data` directory to the `WmRosettaNet\import` directory.
- 3 Import the contents of the copied process archive file into the Trading Networks database. For information about importing PIP archives, see [Chapter 5, “Importing PIP Archives”](#) in this guide.
- 4 Restart the Integration Server.
- 5 Run the PIP 3D8 Sample:
 - a On the Server Administrator navigation panel, under the **Adapters** heading, click **RosettaNet**.
 - b On the Server Administrator navigation panel, click **PIP 3D8 Sample**.
 - c On the PIP 3D8 Large Document Sample page, click **Send Pip3D8**.

Index

A

- ActionCode parameter setting 59
- activating
 - enterprise profiles 32
 - trading partners' profiles 35
- Agreement Details screen
 - Agreement ID field 56
 - Data Status field 56
 - IS Document Type field 56
 - Receiver field 56
 - Sender field 56
- annotations
 - defined 74
- archiving RNOs for auditing 89
- assigning the service that a step invokes 81

C

- components
 - design-time
 - Integration Server 19
 - Modeler 19
 - Modeler Repository 19
 - PIP Archive 18
 - Trading Networks 18
 - Trading Networks Database 18
 - webMethods RosettaNet Module 18
 - run-time
 - Integration Server 22
 - Monitor 22
 - Process Logging Database 22
 - process run time 22
 - Trading Networks 21
 - Trading Networks Database 21
 - webMethods RosettaNet Module 21
- contact information
 - enterprise profiles 31
 - trading partners' profiles 35
- conventions in this guide 2
 - program code 4

- typographical 2
- conversation
 - errors
 - when a business document is out of sequence 97
 - when a transaction times out: notification of failure 96
 - rejoining on the fulfiller's side 86
 - rejoining on the initiator's side 87
 - starting and running 28, 86
 - status
 - sources of 87
 - when errors or exceptions might occur 94
 - why you monitor 87
- ConversationID
 - focal role query 85
 - initiator's D-U-N-S Number query 85
 - process instance identifier query 85
- ConversationID and running a conversation 84
- creating
 - inbound mapping services 67
 - flow operations to use 68
 - input/output to use 68
 - outbound mapping services 64
 - flow operations to use 65
 - input/output to use 64
- custom PIPs
 - implementing 42
- customizing
 - IS document types for existing standard RosettaNet PIPs 41
 - large business documents
 - separating the RosettaNet PIP document into non-repeating and repeating parts 101
 - the inbound mapping service 105
 - the outbound mapping service 103
 - the validation service 106
 - PIP archives 41
 - process model templates 14, 28, 74, 78
 - prerequisites for 78
 - to enable processing of large business documents 101
 - Trading Partner Agreements 27

D

defining

enterprise profiles

required profile fields 31

security information 32

trading partners' profiles 26, 33

required profile fields 33

security information 34

delivery method information

enterprise profiles 32

trading partners' profiles 34

design-time components

Integration Server 19

Modeler 19

Modeler Repository 19

PIP Archive 18

Trading Networks 18

Trading Networks Database 18

webMethods RosettaNet Module 18

determining the process model template to use 78

DTD parameter setting 59

E

Encoding parameter setting 59

Encryption parameter setting 59

EncryptionAlgorithm parameter setting 59

EncryptionHeader parameter setting 59

enterprise profiles

activating 32

defining 31

contact information 31

delivery method information 32

required profile fields 31

security information 32

error handler services

writing 27

error process

initiating manually 97

Error step 77

errors

handling 92

when a business document is out of sequence 97

when a transaction times out: notification of failure 96

when they might occur in a conversation 94

where to find information about 98

with RosettaNet PIP documents 95

example of

inbound mapping service: fulfiller/receiver 70

inbound mapping service: initiator/sender 69

mapping a business document 63

outbound mapping service: fulfiller/receiver 67

outbound mapping service: initiator/sender 66

Exception Handler step 77

exceptions

handling 92

when they might occur in a conversation 94

exporting PIP archives from the Integration Server 42

F

features of the webMethods RosettaNet module 15

figure

commonPIPRequest Non-Repeating Section 102

Create PIP Export Archive Page 43

Example of an Inbound Mapping Service: Fulfiller/ Receiver 70

Example of an Inbound Mapping Service: Initiator/Sender 69

Example of an Outbound Mapping Service: Fulfiller/Receiver 67

Example of an Outbound Mapping Service: Initiator/Sender 66

Example of Mapping a Business Document 63

Inbound Validation of a Large Business Document 110

Initiator/Buyer Process Overview of PIP 3A4 23

Outbound Mapping of a Large Business Document 109

Outbound Mapping Service that Handles Large Business Documents 104

PIP Import Files Page 40

PIP Records Available for Mapping 65

Running a PIP 3A4 Conversation 86

Sample Process Model 74

TN Roles 76

webMethods RosettaNet Module Design-Time Architecture/Components 17

webMethods RosettaNet Module Run-Time Architecture/Components 20

workInProgress Repeating Section 103

focal role 76

focal role query 85

fulfiller

rejoining a conversation 86

G

generating and updating your process model 28

groups

defined 74

H

handler service

defined 92

handler services

wm.ip.cm.handlers:defaultHandler 92

wm.ip.cm.handlers:end 92

wm.ip.cm.handlers:error 92

wm.ip.cm.handlers:inboundValidation 92

wm.ip.cm.handlers:processNOF 93

wm.ip.cm.handlers:processUnhandledDocument 93

wm.ip.cm.handlers:send 93

wm.ip.cm.handlers:sendReceiptACK 93

wm.ip.cm.handlers:sendSyncResponse 93

wm.ip.cm.handlers:start 93

wm.ip.cm.handlers:startNOF 94

wm.ip.cm.handlers:verifyAcknowledgement 94

writing 27

handling errors and exceptions 92

HashAlgorithm parameter setting 57

I

IData objects limitation and large business documents 100

implementing

custom PIP 42

multiple versions of a PIP 15

RosettaNet PIP 26

single PIP 14

importing

PIP archives into the Integration Server 26, 39

Inbound Map step 77

inbound mapping service: fulfiller/receiver

example of 70

inbound mapping service: initiator/sender

example of 69

inbound mapping services

creating 67

flow operations to use 68

input/output to use 68

customizing for large business documents 105

handling large business documents 105

why you create 62

writing 27

inbound validation of a large business document 110

initiating an error process manually 97

initiator

rejoining a conversation 87

initiator's D-U-N-S Number query 85

installing the webMethods RosettaNet module packages 26

introduction to this guide 2

L

large business document sample 111

large business documents 101

customizing

separating the RosettaNet PIP document into non-repeating and repeating parts 101

the inbound mapping service 105

the outbound mapping service 103

the validation service 106

how an inbound mapping service handles 105

limits to processing 100

outbound mapping of 109

run-time processing of 108

the IData objects limitation 100

the pipeline limitation 100

limits to processing large business documents 100

location of the process model templates 76

M

mapping a business document

defined 62

mapping services

reusing 70

modifying the Sender, Receiver, and Agreement ID 54

Monitor 88

N

NSFolder parameter setting 59

O

Outbound Map step 77
outbound mapping of a large business document 109
outbound mapping service: fulfiller/receiver
 example of 67
outbound mapping service: initiator/sender
 example of 66
outbound mapping services
 creating 64
 flow operations to use 65
 input/output to use 64
 customizing for large business documents 103
 why you create 62
 writing 27
overview of RosettaNet process 23

P

packages
 WmIPRoot 16
 WmRNIF11TRP 16
 WmRNIF20TRP 16
 WmRNLargeDocSample 17
 WmRNPIps 16
 WmRNSample 17
 WmRosettaNet 16
parameter settings 57
 ActionCode 59
 DTD 59
 Encoding 59
 Encryption 59
 EncryptionAlgorithm 59
 EncryptionHeader 59
 HashAlgorithm 57
 NSFolder 59
 ProcessCode 58
 ProcessDescription 58
 ProcessVersion 58
 RC2EncryptionKeyLength 59
 ReceiverClassification 59

SenderClassification 59
Sign 57
Signature Required 57
ThirdPartyPayloadType 59
ThirdPartyPayloadVersion 60
ThirdPartyReferenceId 60
TransactionCode 58
Transport 57
TransportVersion 57
UsageCode 59
ValidateOutput 58
ValidationService 58
PIP archives 18
 contents of
 IS document types 38
 TN XML document types 38
 Trading Partner Agreements 38
 customizing 41
 defined 38
 exporting from the Integration Server 42
 importing into the Integration Server 26, 39
 why you need to import 38
pipeline limitation and large business documents 100
PipInfo TPA section 57
PIPs
 and process models 39
 implementing 14
 custom PIP 42
 types of
 one-action asynchronous 39
 two-action asynchronous 39
 two-action synchronous 39
prerequisites for customizing process model templates 78
printing this guide 11
Process ACK step 77
process instance identifier query 85
process model templates
 and focal role 76
 and TN roles 75
 customizing 14, 28, 74, 78
 determining which one to use 78
 location of 76
 provided by webMethods 76

- steps
 - Error 77
 - Exception Handler 77
 - Inbound Map 77
 - Outbound Map 77
 - Process ACK 77
 - Send to Back-End 77
 - Wait for External Request Document 77
 - Wait for External Response Document 77
 - Wait for Internal Request Document 77
 - Wait for Internal Response Document 77
 - steps you must modify 77
 - process models
 - annotations 74
 - defined 74
 - generating and updating 28
 - groups 74
 - steps 74
 - transitions 74
 - process models and PIPs 39
 - process overview 23
 - process run time
 - and process model steps 74
 - defined 14
 - ProcessCode parameter setting 58
 - ProcessDescription parameter setting 58
 - ProcessInfo TPA section 57
 - ProcessVersion parameter setting 58
 - program code conventions 4
- Q**
- query
 - focal role 85
 - initiator's D-U-N-S Number 85
 - process instance identifier 85
 - ReceiverID 51
 - SenderID 51
- R**
- RC2EncryptionKeyLength parameter setting 59
 - ReceiverClassification parameter setting 59
 - ReceiverID query 51
 - rejoining an existing conversation
 - on the fulfiller's side 86
 - on the initiator's side 87
 - related documentation 4
 - retry count values
 - setting the 81
 - reusing mapping services 70
 - RNIF2.0 TPA section 57
 - RosettaNet
 - conversation
 - when errors or exceptions might occur 94
 - process overview 23
 - RosettaNet PIP
 - implementing 26
 - RosettaNet PIP documents
 - general errors with 95
 - running a conversation
 - and the ConversationID 84
 - example 85
 - wm.ip.cm:processDocument service 85
 - wm.ip.rn:receive service 85
 - running the PIP 3D8 sample 111
 - run-time components
 - Integration Server 22
 - Monitor 22
 - Process Logging Database 22
 - process run time 22
 - Trading Networks 21
 - Trading Networks Database 21
 - webMethods RosettaNet Module 21
- S**
- sample
 - large business document 111
 - Send to Back-End step 77
 - SenderClassification parameter setting 59
 - SenderID query 51
 - services
 - wm.ip.cm:processDocument 85
 - wm.ip.rn:receive 85
 - setting timeout and retry count values 81
 - settings
 - TPA parameter 57
 - ActionCode 59

- DTD 59
- Encoding 59
- Encryption 59
- EncryptionAlgorithm 59
- EncryptionHeader 59
- HashAlgorithm 57
- NSFolder 59
- ProcessCode 58
- ProcessDescription 58
- ProcessVersion 58
- RC2EncryptionKeyLength 59
- ReceiverClassification 59
- SenderClassification 59
- Sign 57
- SignatureRequired 57
- ThirdPartyPayloadType 59
- ThirdPartyPayloadVersion 60
- ThirdPartyReferenceld 60
- TransactionCode 58
- Transport 57
- TransportVersion 57
- UsageCode 59
- UseProfileCerts 58
- ValidateOutput 58
- ValidationService 58
- Sign parameter setting 57
- SignatureRequired parameter setting 57
- sources of conversation status and information 87
- starting and running a conversation 28, 86
- steps
 - defined 74
 - process model templates
 - Error 77
 - Exception Handler 77
 - Inbound Map 77
 - Outbound Map 77
 - Process ACK 77
 - Send to Back-End 77
 - Wait for External Request Document 77
 - Wait for External Response Document 77
 - Wait for Internal Request Document 77
 - Wait for Internal Response Document 77
 - steps in the process model templates that you must modify 77

T

- templates
 - customizing 14, 28, 74, 78
 - determining which one to use 78
 - location of 76
 - prerequisites for customizing 78
 - provided by webMethods 76
 - steps
 - Error 77
 - Exception Handler 77
 - Inbound Map 77
 - Outbound Map 77
 - Process ACK 77
 - Send to Back-End 77
 - Wait for External Request Document 77
 - Wait for External Response Document 77
 - Wait for Internal Request Document 77
 - Wait for Internal Response Document 77
 - steps you must modify 77
- ThirdPartyPayloadType parameter setting 59
- ThirdPartyPayloadVersion parameter setting 60
- ThirdPartyReferenceld parameter setting 60
- timeout values
 - setting the 81
- TN roles 75
- TN XML document types
 - defined 48
- TPA
 - distinguishing between an initiator's and a fulfiller's 57
 - identifying 54
 - modifying the Sender, Receiver, and Agreement ID 54
 - parameter settings 57
 - ActionCode 59
 - DTD 59
 - Encoding 59
 - Encryption 59
 - EncryptionAlgorithm 59
 - EncryptionHeader 59
 - HashAlgorithm 57
 - NSFolder 59
 - ProcessCode 58
 - ProcessDescription 58
 - ProcessVersion 58

- RC2EncryptionKeyLength 59
- ReceiverClassification 59
- SenderClassification 59
- Sign 57
- SignatureRequired 57
- ThirdPartyPayloadType 59
- ThirdPartyPayloadVersion 60
- ThirdPartyReferenceId 60
- TransactionCode 58
- Transport 57
- TransportVersion 57
- UsageCode 59
- UseProfileCerts 58
- ValidateOutput 58
- ValidationService 58
- sections
 - PipInfo 57
 - ProcessInfo 57
 - RNIF2.0 57
- Trading Networks
 - function of 18
- Trading Partner Agreements
 - customizing 27
 - defined 14, 54
- trading partners
 - defined 30
- trading partners' profiles
 - activating 35
 - defining 26, 33
 - contact information 33
 - delivery method information 34
 - required profile fields 33
 - security information 34
 - why they are important 30
- TransactionCode parameter setting 58
- transitions
 - defined 74
- Transport parameter setting 57
- TransportVersion parameter setting 57
- typographical conventions 2

U

- UsageCode parameter setting 59

- UseProfileCerts parameter setting 58

V

- ValidateOutput parameter setting 58
- validation
 - inbound 110
- validations service
 - customizing for large business documents 106
- ValidationService parameter setting 58
- viewing this guide 11

W

- Wait for External Request Document step 77
- Wait for External Response Document step 77
- Wait for Internal Request Document step 77
- Wait for Internal Response Document step 77
- webMethods RosettaNet module
 - advantages and disadvantages of customizing and exporting
 - PIP archives 45
 - defined 14
 - design-time components
 - Integration Server 19
 - Modeler 19
 - Modeler Repository 19
 - PIP Archive 18
 - Trading Networks 18
 - Trading Networks Database 18
 - webMethods RosettaNet Module 18
 - features 15
 - handler services provided with 92
 - identifying a TPA 54
 - installing 26
 - packages
 - WmIPRoot 16
 - WmRNIF11TRP 16
 - WmRNIF20TRP 16
 - WmRNLargeDocSample 17
 - WmRNPips 16
 - WmRNSample 17
 - WmRosettaNet 16
 - run-time components
 - Integration Server 22
 - Monitor 22

- Process Logging Database 22
- process run time 22
- Trading Networks 21
- Trading Networks Database 21
- webMethods RosettaNet Module 21
- wm.ip.cm.handlers:defaultHandler handler service 92
- wm.ip.cm.handlers:end handler service 92
- wm.ip.cm.handlers:error handler service 92
- wm.ip.cm.handlers:inboundValidation handler service 92
- wm.ip.cm.handlers:processNOF 93
- wm.ip.cm.handlers:processUnhandledDocument handler service 93
- wm.ip.cm.handlers:send handler service 93
- wm.ip.cm.handlers:sendReceiptACK handler service 93
- wm.ip.cm.handlers:sendSyncResponse handler service 93
- wm.ip.cm.handlers:start handler service 93
- wm.ip.cm.handlers:startNOF 94
- wm.ip.cm.handlers:verifyAcknowledgement handler service 94
- wm.ip.cm:processDocument service
 - running a conversation 85
- wm.ip.rn:receive service
 - running a conversation 85
- WmIPRoot package 16
- WmRNIF11TRP package 16
- WmRNIF20TRP package 16
- WmRNLargeDocSample package 17
- WmRNPips package 16
- WmRNSample package 17
- WmRosettaNet package 16
- writing
 - error handlerservices 27
 - inbound and outbound mapping services 27