webMethods RosettaNet Module Installation and User's Guide

Version 7.1 SP2

October 2012

WEBMETHODS

This document applies to webMethods RosettaNet Module Version7.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ESTD-RN-IUG-71SP2-20160107

Table of Contents

About this Guide

This guide describes how to implement and monitor the exchange of RosettaNet Partner Interface Processes (PIPs) using the webMethods RosettaNet Module. It contains information for administrators and application developers who want to exchange data, such as RosettaNet business messages, with other trading partners using the RosettaNet implementation framework (RNIF).

To use this guide effectively, you should be familiar with:

- webMethods Integration Server, Integration Server Administrator, and the concepts and procedures described in the Integration Server administration guide for your release version.
- webMethods Trading Networks and the concepts and procedures described in the Trading Networks guides for your release version.
- Software AG Designer and the concepts and procedures described in the *Software AG Designer Online Help*.
- webMethods Monitor and the concepts and procedures described in the Monitor guides for your release version.
- webMethods Process Engine and the concepts and procedures described in the Process Engine guides for your release version.
- Have a basic knowledge of the RosettaNet standard and related RosettaNet terminology.

Deprecation of webMethods Developer

webMethods Developer is deprecated and does not support all the features of webMethods Integration Server 8.2. SoftwareAG recommends the use of SoftwareAG Designer for service development.

Document Titles

Some webMethods document titles have changed during product releases. The following table will help you locate the correct document for a release on the Software AG Documentation Web site or the Empower Product Support Web site.

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
Designer Process Development online help					
webMethods BPM Process Development Help				x	x
webMethods Designer BPM Process Development Help		x	x		
webMethods Designer Process Development Help	x				
Designer Service Development online help					
webMethods Service Development Help				x	x
webMethods Designer Service Development Help	x	x	x		
Developer user's guide					
Developing Integration Solutions: webMethods Developer User's Guide			x	x	
webMethods Developer User's Guide	x	x			
Integration Server administration guide					
webMethods Integration Server Administrator's Guide	x	x			x
Administering webMethods Integration Server			x	x	
Integration Server built-in services reference guide					
webMethods Integration Server Built-In Services Reference	x	x	x	x	x
Integration Server clustering guide					

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
webMethods Integration Server Clustering Guide	x	x	x	x	x
Integration Server publish-subscribe developer's guide					
Publish-Subscribe Developer's Guide	x	x	x	x	x
My webMethods administration guide					
Administering My webMethods Server			x	x	x
My webMethods Server Administrator's Guide	x	x			
Optimize administration guide					
Administering webMethods Optimize			x	x	x
webMethods Optimize Administrator's Guide	x	x			
Optimize user's guide					
webMethods Optimize User's Guide	x	x			x
Optimizing BPM and System Resources with BAM: webMethods Optimize User's Guide			x	x	
Process Engine administration guide					
Administering webMethods Process Engine		x	x	x	x
webMethods Process Engine User's Guide	x				
Trading Networks administration guide					
webMethods Trading Networks Administrator's Guide	x	x			x

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
Building B2B Integrations: webMethods Trading Networks Administrator's Guide			x	x	
Trading Networks built-in services reference guide					
webMethods Trading Networks Built-In Services Reference	x	x	x	x	x
Trading Networks concepts guide					
webMethods Trading Networks Administrator's Guide and *webMethods Trading Networks User's Guide*					x
Understanding webMethods B2B: webMethods Trading Networks Concepts Guide			x	x	
webMethods Trading Networks Concepts Guide	x	x			
Trading Networks user's guide					
webMethods Trading Networks User's Guide	x	x			x
Managing B2B Integrations: webMethods Trading Networks User's Guide			x	x	
webMethods installation guide					
Installing webMethods Products and *Using the Software AG Installer*				x	x
Software AG Installation Guide			x		
webMethods Installation Guide	x	x			
webMethods logging guide					
webMethods Audit Logging Guide			x	x	x

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
webMethods Audit Guide	x	x			
webMethods upgrade guide					
Upgrading webMethods Products				x	x
webMethods Upgrade Guide	x	x	x		

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.

Convention	Description
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

1 Concepts

What Is webMethods RosettaNet Module?

webMethods RosettaNet Module is an implementation of the RosettaNet Implementation Framework (RNIF), versions 1.1 and 2.0. webMethods RosettaNet Module enables you to implement open and common e-business Partner Interface Processes (PIPs) and exchange RosettaNet documents with your partners. Business partners use RosettaNet PIPs to execute business-to-business transactions and processes.

PIPs are organized into one of seven categories according to their high-level business area, such as, product information, order management, or inventory management. Sub-categories further define the business segments and process category. Each of these processes define the specific flow of the process model and structure of the business document.

The webMethods RosettaNet Module runs on the webMethods Integration Server platform and integrates with other products in the webMethods product suite, leveraging process modeling and management capabilities, and XML business document recognition capabilities. For RosettaNet Module to exchange messages with your partners, you must configure objects in some of these integrated webMethods components, as explained below.

- **webMethods Trading Networks** for defining your trading partners and handling tasks related to identifying partners and the documents they send.
- **Software AG Designer**for creating new services and processes.

RosettaNet Module integrated with the webMethods Trading Networks enables you to define your trading partners and handle tasks related to identifying partners and the documents they send. This component uses TN XML document types to identify specific documents and Trading Partner Agreements (TPAs) for defining specific processing related to a unique sender and receiver combination of a specific document type.

In RosettaNet Module, every PIP is associated with two Trading Partner Agreements (TPAs), one for the initiator (sender) in a conversation and one for the fulfiller (receiver). A TPA is a set of parameters that govern how business documents are exchanged between two trading partners. For example, a TPA contains parameters that control signing, encoding, and validation of RosettaNet PIP documents. For more information about using TPAs in RosettaNet Module, see "Customizing Trading Partner Agreements" on page 61. For detailed information about TPAs, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

You can implement a PIP by importing the necessary *PIP archives*, if available, or by importing the RosettaNet standard provided PIP specification zip file using webMethods PIP Tools, and then customizing one of the eight model templates as described in "Customizing Process Model Template" on page 79

You can use Software AG Designer for modifying process model templates and creating custom services specific to your business needs. When you customize a process model template in Designer, you can assign custom-created services to individual steps in the

template and assign TN XML document types to *receive steps*. *Receive steps* are steps that "wait" for a specific incoming business document during a conversation (also called a process) so that RosettaNet Module can recognize the document type and ensure that documents are exchanged and processing proceeds according to an expected and predictable process flow. For information about customizing process model templates, see "Customizing Process Model Template" on page 79. For more information about process models and Software AG Designer, see the Designer online help for your release. See "About this Guide" for specific document titles.

webMethods Process Engine is a component of Integration Server that manages the execution of RosettaNet conversations. Process Engine is the process model enforcer, essentially ensuring that conversations are executed as defined by the process model, executing process logic, logging process data, and controlling process order execution. During run time, Process Engine uses the *ConversationID* associated with a business process to determine whether the process is active and currently running. For more information about Process Engine, see " RosettaNet Module Run-Time Architecture and Components" on page 25. For more information about *ConversationID* s, see "How Monitor Tracks Conversations and the ConversationID" on page 92. For detailed information about Process Engine, see *Administering webMethods Process Engine*. See "About this Guide" for specific document titles.

RosettaNet Module consists of a set of design-time and run-time components, discussed in this chapter. For information about design-time components, see " RosettaNet Module Design-Time Architecture and Components" on page 23. For information about run-time components, see " RosettaNet Module Run-Time Architecture and Components" on page 25.

RosettaNet Module Features

RosettaNet Module runs on Integration Server and provides the following features and capabilities:

- **Supports the RNIF and RosettaNet PIPs.** Quickly implement production solutions for automating the many interactions across your entire supply chain. 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*, provided by RosettaNet to enhance the RNIF standard.
- **Implement new PIP versions quickly.** Stay current with new versions from RosettaNet, without waiting for a new software release.
- **Use custom PIPs.** Unlike other solutions, RosettaNet Module does not require "hard-coding" of PIPs. With RosettaNet Module, you can implement custom PIPs if needed.
- **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 might 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.

- **Capture trading partner-specific rules.** Rules that reflect the unique business practices between your organization and your trading partners.
- **Leverage existing investments in enterprise solutions.** Exchange information from EDI-based systems to populate business documents in RosettaNet format.
- **Customized RosettaNet-based process models.** Use Designer to create business process models that reflect your organization's business requirements.
- **Monitor your process models.** Use Monitor to manage and monitor your process models.
- **Maintain 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.
- **Synchronous/Asynchronous transmission of PIP messages.** With RosettaNet Module, you can employ either synchronous or asynchronous communication to meet the time requirements of your trading partner transactions. For those PIPs that require immediate responses, you can use synchronous communication. For transactions with longer duration, you can use asynchronous communication.
- **Supports multi-byte languages.** Multiple language support includes languages such as Chinese and Japanese.
- **Process large business documents.** Use Integration Server's services to process large business documents.

RosettaNet Module Solution Overview

This section provides a high-level overview of the primary components and configurations that you must complete to process RosettaNet transactions.

Configuring Trading Networks Assets

RosettaNet Module integrated with Trading Networks allows you to create a business-to-business trading network for exchanging messages. Trading Networks has several "assets" that you must configure. Trading Networks uses these assets to identify RosettaNet documents and determine how to process them. Using Trading Networks's user interface, My webMethods, configure the following assets during design time:

- Trading Networks*profiles* define each organization that exchanges messages on your network. Define a profile for your organization, known as the *enterprise* profile, and a profile for each partner or organization with whom you exchange messages.

- *TN XML document types* identify the types of documents that you and your partners exchange. Each document type represents a specific business purpose and defines a specific format to which the message must adhere. TN XML document types consist of attributes that identify the data that must be populated in the message. When you define the document type, you specify which attributes (data) are required and which to extract (for incoming message) or populate (for outbound messages).

  Trading Networks matches each document it receives to an (enabled) TN XML document type, which indicates how to process the message or transform it into the format required by an internal system or into a RosettaNet message.

  Each document should match only one TN document type. If Trading Networks finds multiple TN XML document types that match a document, the module cannot determine which document type to use and stops processing the document with an error.

  If a PIP is imported using PIP Tools, the required TN XML document types for that PIP are created automatically.

- *Trading Partner Agreements (TPAs)* are a set of parameters that define how business documents are exchanged between trading partners, and define specific parameters for signing, encoding, and validating RosettaNet documents. The TPA allows you to narrow Trading Networks document processing instructions to a specific partner and message type combination. The TPA includes the following:

  - Two Trading Networks profiles, one assigned as the sender, and one assigned as the receiver.
  - One TN XML document type assigned as the agreement ID.

  If a PIP is inserted using PIP Tools, the required TPAs are created automatically, with both Sender and Receiver configured as Unknown.

For information about defining trading partner profiles, see "Defining Trading Partner Profiles " on page 41. For information about defining TN XML document types, see "Defining Internal TN XML Document Types" on page 57. For information about defining TPAs, see "Customizing Trading Partner Agreements" on page 61. For complete information about using Trading Networks and Trading Networks assets, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Implementing Process Models

RosettaNet Module allows you to create process models to completely customize document processing. A process model visually depicts the individual steps and system interactions of a business process, using the Trading Networks assets that you create. RosettaNet Module provides process templates that you can modify to match your own system processes.

RosettaNet Module process templates are at *Integration Server_directory*\packages\WmRosettaNet\ProcessModels directory. The elements required to implement a specific RosettaNet process includes IS document types, TN XML document types

and Trading Partner Agreements. You can create these elements using PIP Tools, after importing the required PIP specification zip file (provided by RosettaNet Standard) via PIP Tools. You can configure individual process steps that are triggered by a certain event, for example, a *receive step* that stops processing until it receives a specific business document (such as an acknowledgment) from your partner or a *wait step*, that executes the steps to process a document. Wait steps are dependent on the successful execution of receive steps; a wait step must follow a receive step. For more information about wait steps, refer to the pub.estd.rosettaNet:waitStepInit service.

Designer is the tool you use to import and configure business process. Process Engine executes the appropriate process model for a given RosettaNet transaction, and monitors the flow of conversations, ensuring that correct documents are exchanged and in the right order. Process Engine works with Monitor to store all conversation and transactional data, while Trading Networks identifies documents, processes them based on the sender and document type, executes any processing rules, and invokes any assigned transformation services.

For specific details about configuring process models in RosettaNet Module, see "Customizing Process Model Template" on page 79. For more information about RosettaNet Process Archives, see "Importing PIP Specification/PIP Archives" on page 49. For detailed information about process models and Designer, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.

Processing Overview

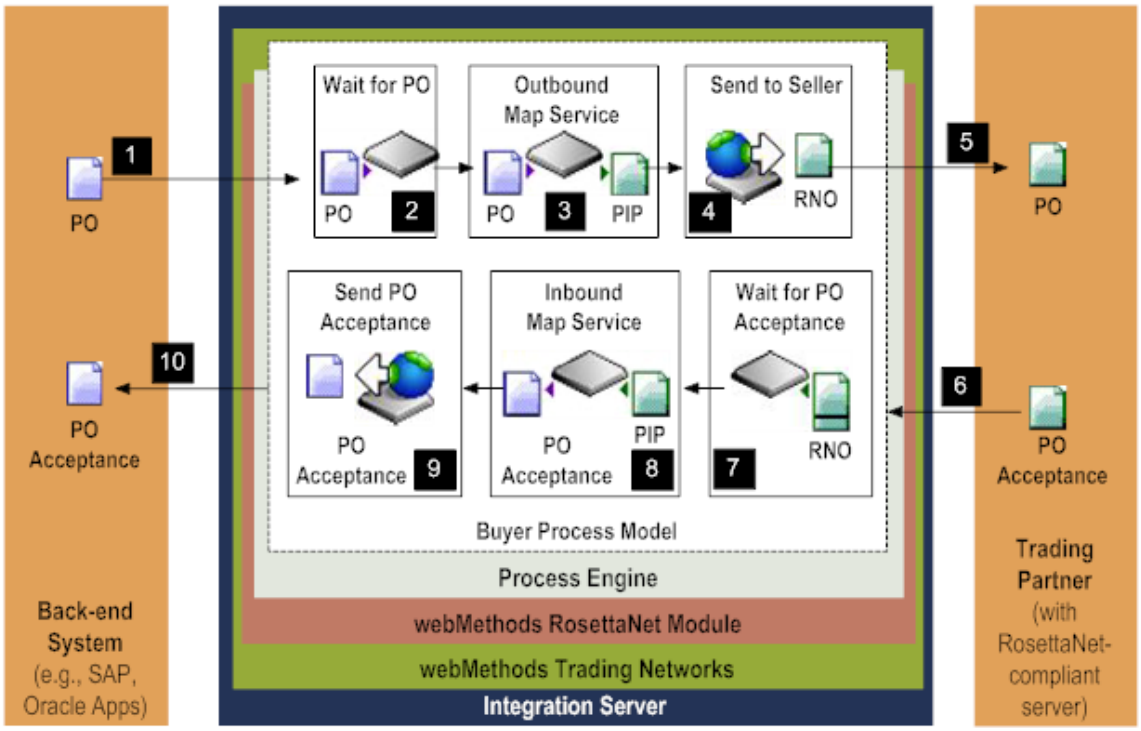
RosettaNet Module divides processing of business messages among the components of the RosettaNet Module solution. RosettaNet Module passes all documents (outbound and incoming) to Trading Networks. Once processing is completed by Trading Networks, RosettaNet Module passes the processed documents to Process Engine.

Process Engine determines which process model to use for a RosettaNet transaction and tracks the document processing using the *ConversationID* . Process Engine determines if the *ConversationID* belongs to a new or existing conversation:

- If the *ConversationID matches* a running process instance, it rejoins the document to the running conversation.
- If the *ConversationID does not match* a running process instance, Process Engine starts a new conversation using the appropriate process model.

Process Engine tracks the processing status in the Process Audit Log database.

The following example of document processing illustrates the RosettaNet *PIP 3A4 Request Purchase Order*process to send a purchase order (PO) to a trading partner. Only the conversation for the buyer role is displayed.



Step	Description
1	Enterprise is the initiator (buyer). Your internal (back-end) system generates a purchase order (PO) and submits it to RosettaNet Module. RosettaNet Module receives the PO and invokes a Trading Networks service to match the document to a TN XML document type. RosettaNet Module wraps the PO in a BizDocEnvelope and saves the BizDocEnvelope in the database using Trading Networks service.
2	RosettaNet Module passes the BizDocEnvelope to Process Engine. Process Engine searches for a running process instance with a matching *ConversationID* . If Process Engine does not find one, it starts a new conversation (process). At the Receive PO step, a service retrieves the Trading Networks profiles for the sender, 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 (the seller).
5	Now on the receiver's side, the trading partner receives the PO, validates it, and sends a Receipt Acknowledgment (not illustrated).

Step	Description
6	The trading partner returns a PO Acceptance as a RNO, sent to your enterprise through RosettaNet Module. Trading Networks receives the PO Acceptance and extracts the *ConversationID* . Trading Networks then passes the PO Acceptance to Process Engine.
7	Process Engine continues the conversation by matching the *ConversationID* from the PO Acceptance with the *ConversationID* in the already running process instance. Once the conversation resumes, the PO Acceptance is validated, and your enterprise sends a Receipt Acknowledgment (not illustrated) to the trading partner.
8	Process Engine invokes an inbound mapping service which maps the PO Acceptance from a PIP to the format used by your internal system.
9	At the **Send PO Acceptance** step, a custom-created service sends the PO Acceptance to your internal system.
10	Your internal system receives the PO Acceptance.

For more information about Process Engine, see " RosettaNet Module Run-Time Architecture and Components" on page 25 and *Administering webMethods Process Engine*. For more information about *ConversationIDs* , see "How Monitor Tracks Conversations and the ConversationID" on page 92.

RosettaNet Module Packages

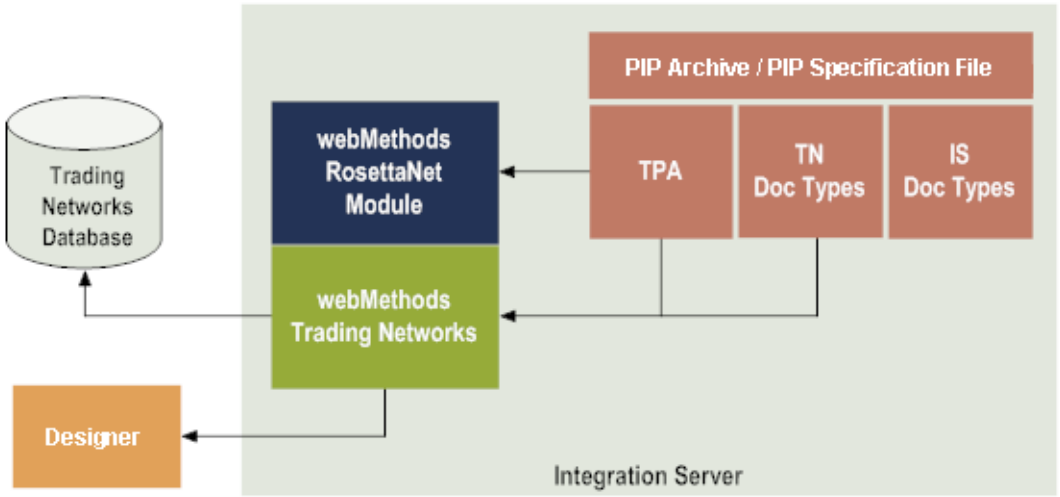
RosettaNet Module contains several packages (sets of webMethods services and related files) to be installed on the Integration Server. The following table describes the contents of each package. For detailed information about the contents of the public services in WmRosettaNet package, see "Public Services in WmRosettaNet Package" on page 93.

Package	Description
WmRosettaNet	Contains the common RosettaNet functionality required by the WmRNIF11TRP and WmRNIF20TRP packages. For more information about the WmRosettaNet package, see "Public Services in WmRosettaNet Package" on page 93.
WmRNIF11TRP	Contains the functionality specific to the implementation of RNIF version 1.1 transport protocol.

Package	Description
WmRNIF20TRP	Contains the functionality specific to the implementation of RNIF version 2.0 transport protocol.
WmRNPips	Contains the records for the specific PIPs that you have imported into RosettaNet Module. **Note:** This package is created automatically when you import the first PIP.

RosettaNet Module Design-Time Architecture and Components

RosettaNet Module consists of a set of design-time and run-time components. The following diagram illustrates the design-time architecture in RosettaNet Module. The table following the diagram explains each of these components.

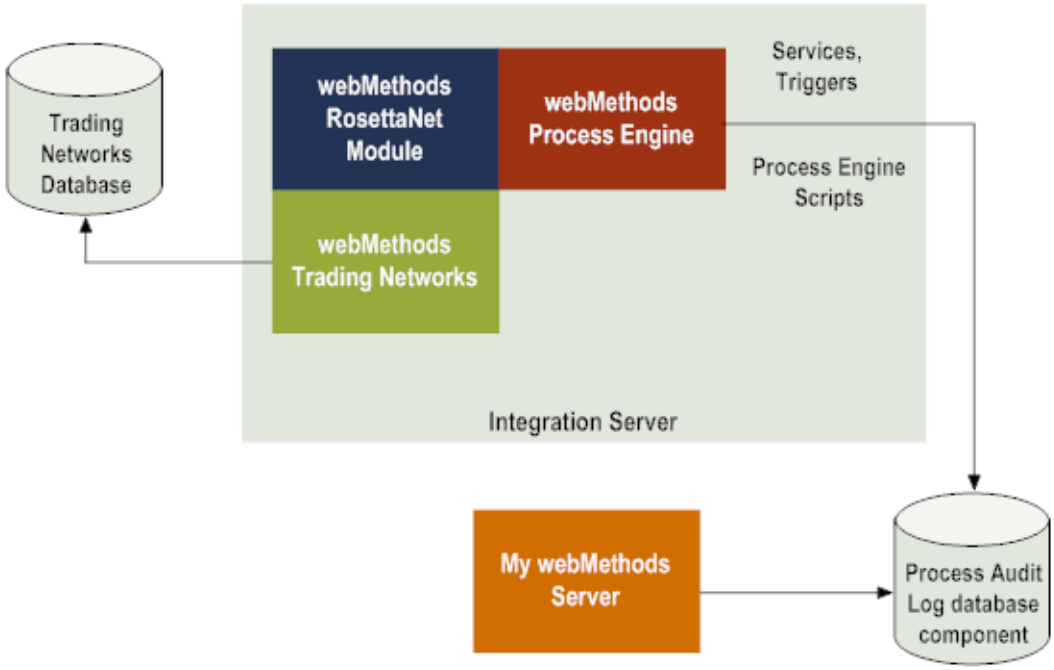


Component	Description
PIP Archive/PIP Specification File	A PIP Specification File / Partner Interface Process (PIP) archive file contains the following: ■ Two Trading Partner Agreements (TPAs): one for the initiator (sender) and one for the fulfiller (receiver) in a conversation ■ TN XML document types ■ IS document types for a specific version of a particular PIP

Component	Description
	During design time, import the PIP archives you will use into Integration Server.
RosettaNet Module	RosettaNet Module contains process model templates for the different types of PIPs that you can implement. You can use Designer to modify these process model templates as needed to create your PIPs.
Trading Networks	Trading Networks links you and your trading partners to form a business-to-business trading network. Trading Networks uses trading partner profiles to determine how to exchange business documents with your trading partners. During design time, you define your trading partner profiles through My webMethods (the interface to Trading Networks). You can also customize the Trading Partner Agreements (TPAs) and view TN XML document types. TPAs and TN XML document types are created in Trading Networks when you import the PIP using PIP Tools or by importing a PIP archive file into Integration Server.
Trading Networks Database	Trading Networks saves trading partner profiles TN XML document types, and TPA information in its database and retrieves this information when needed.
Designer	Designer allows you to create process models and customize the process model templates provided with RosettaNet Module. When you customize a process model template, you can specify how the process model interacts with your back-end (internal) systems and specify the services to be invoked during individual steps in the process. When you build and upload a process model, the run-time elements (services, triggers, and Process Engine scripts, or fragments) are generated automatically. For more information about Designer, see the Designer online help. For information about customizing a process model template, see "Customizing Process Model Template" on page 79.
Integration Server	Integration Server contains the documents, services, and records that you can access when creating your process models.

RosettaNet Module Run-Time Architecture and Components

The following diagram illustrates RosettaNet Module run-time architecture. The table following the diagram explains each of these components.



Component	Description
RosettaNet Module	During run time, RosettaNet Module receives a business document from an internal (back-end) system or trading partner. RosettaNet Module integrates with Trading Networks to execute document exchange as described in the next row. RosettaNet Module passes the business document to Process Engine.
Trading Networks and My webMethods	Trading Networks links you with your trading partners to form a business-to-business trading network. My webMethods is the user interface to Trading Networks. Trading Networks services do the following: ■ Recognize incoming business documents. ■ Create BizDocEnvelopes. ■ Save BizDocEnvelopes to the Trading Networks database.

Component	Description
	Trading Networks uses trading partner profiles to determine how to exchange business documents with your trading partners. Trading Networks TPAs define information such as whether an outbound business document should be signed or whether the service header in the business document should be encrypted.
Trading Networks Database	The Trading Networks database stores TN XML document type, TPA, trading partner profile information and RosettaNet transactions.
My webMethods Server	My webMethods Server is a run-time container for functions made available by webMethods applications. The user interface in which you perform these functions is called My webMethods. My webMethods provides a ready-made environment in which users can perform functions on webMethods applications, and administrators can manage access to those functions. In addition, My webMethods Server gives you the capability to develop additional user interface pages, and a broad-based set of administrative tools with which to manage the increased capabilities.
webMethods Process Engine	Process Engine is a facility of Integration Server that manages the execution of RosettaNet conversations. Process Engine ensures the integrity, traceability, controllability and ability to observe RosettaNet conversations by performing the following functions: ■ Accepting business documents from Trading Networks. ■ Determining which process model to apply to the PIP transaction. ■ Processing a business document based on the sender and the document type and *ConversationID* . ■ Recording the status of the business document in the Process Audit Log database. To process a business document, Process Engine uses the document's *ConversationID* to determine whether the document belongs to a new or existing conversation and matches incoming documents to the right conversation.
Monitor	Monitor displays data that Integration Server has logged. Monitor displays data about business processes, services, documents, and conversations, retrieved from the Process

Component	Description
	Audit Log database. Use Monitor to manage and monitor conversations. You can also manage a conversation with commands such as suspend, resume, restart, and stop. My webMethods is the user interface to Monitor.
Process Audit Log Database	Monitor and Process Engine log audit data about running conversations to the Process Audit Log database.
Integration Server	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 Engine scripts (or fragments).

2 Installing webMethods RosettaNet Module

Overview

This chapter, along with the webMethods installation guide for your release, explains how to install, upgrade, and uninstall RosettaNet Module 7.1 SP2. See "About this Guide" for specific document titles.

Important: For the webMethods 8 release, use Software AG Installer 8 to install the module and its components. The installer for prior versions of webMethods products was webMethods Installer 7. If you are installing the module with webMethods 8 products, you must use Software AG Installer 8 and the webMethods installation guide for the 8.x release. If you are installing the module with webMethods 7.x, you can still use Software AG Installer 8 with the webMethods installation guide for the 8.x release or use webMethods Installer 7 with *webMethods Installation Guide 7.x*. See "About this Guide" for specific document titles.

Requirements

For a list of the operating systems and webMethods products that RosettaNet Module 7.1 SP2 supports, see *webMethods eStandards Modules System Requirements*. RosettaNet Module 7.1 SP2 has no hardware requirements beyond those of its host Integration Server.

Installing RosettaNet Module 7.1 SP2

The instructions in this section explain how to install RosettaNet Module 7.1 SP2 using the Software AG Installer wizard. These instructions are meant to be used with the more complete instructions in the webMethods installation guide for your release.

Note: If you are installing RosettaNet Module in a clustered environment, you must install it on each Integration Server in the cluster, and each installation must be identical. For more information about working with RosettaNet Module in a clustered environment, see *webMethods Integration Server Clustering Guide*.

To install RosettaNet Module 7.1 SP2

1. Download the Software AG Installer from the Empower Product Support website at https://empower.softwareag.com.
2. If you are installing the module on an existing Integration Server, shut down Integration Server.
3. Start the Software AG Installer wizard and follow the instructions to install RosettaNet Module:

a. In the **Release** list, select the webMethods release that corresponds to the version of Integration Server you are using. For example, if you want to install RosettaNet Module on Integration Server 8.2, choose the 8.2 release.

b. Enter your Software AG Empower user name and password. Installer uses these to connect to the Installer server and populate your licensed products in the selection.

c. Specify the installation directory:

- If you have already installed the Integration Server platform, specify the webMethods installation directory that contains Integration Server.
- If you are installing both Integration Server and the module, specify the installation directory for Integration Server. The installer installs RosettaNet Module within the *Integration Server_directory*\packages directory.

d. In the product selection list, select **eStandards > webMethods RosettaNet Module 7.1 SP2**. Verify that **Program Files** and the required transport version are also selected.

For example, to install the RNIF 1.1 implementation, select **eStandards > webMethods RosettaNet Implementation Framework 7.1 SP2 > RNIF 1.1**.

e. If you want to install PIP Tools, select **eStandards > Common Files 7.1 > PIPTools**.

f. Optionally, install any other required products as indicated in *webMethods eStandards Modules System Requirements*.

The installer installs the following components:

- WmEstdCommonLib (eStandards Common Library)
- WmRosettaNet
- The selected implementation framework, for example, WmRNIF11TRP or WmRNIF20TRP
- WmPIPTools, if PIP Tools is selected

4. Use the Software AG Update Manager located on the Empower Product Support website at https://empower.softwareag.com, to install the latest fixes for RosettaNet Module and other webMethods products. For a list of required fixes, see *webMethods eStandards Modules System Requirements* .

   For information about installing the fixes, see the instructions in the corresponding fix readme file.

5. Restart Integration Server.

   RosettaNet Module 7.1 SP2 starts automatically.

Upgrading to RosettaNet Module 7.1 SP2

This section describes how to upgrade and migrate the services created in:

- RosettaNet Module 7.1 SP1 to RosettaNet Module 7.1 SP2
- RosettaNet Module 7.1 to RosettaNet Module 7.1 SP2
- RosettaNet Module 6.0.1 to RosettaNet Module 7.1 SP2

Before You Begin

Before you upgrade to RosettaNet Module 7.1 SP2, verify that you have installed a compatible version of Integration Server, as well as the required component products such as Trading Networks, Designer and Monitor. Integration Server must be version 7.1.3 or later to run RosettaNet Module 7.1 SP2. For instructions, see the webMethods upgrade guide for your release. See “About this Guide” for specific document titles.

When you are upgrading to a newer version of Trading Networks make sure that you back up the RosettaNet Module packages that contain Trading Networks specific data. For more information on how to upgrade Trading Networks, see the webMethods upgrade guide for your release. See “About this Guide” for specific document titles.

Note: The 6.5.x equivalent for Designer was Modeler.

Upgrading from RosettaNet Module 7.1 and 7.1 SP1

To upgrade from RosettaNet Module 7.1 and RosettaNet Module 7.1 SP1

1. Back up the entire webMethods installation directory where the existing RosettaNet Module is installed on each machine, and your webMethods databases as instructed by your RDBMS vendor.
2. Uninstall the existing RosettaNet Module using the Software AG Uninstaller. For instructions on uninstalling RosettaNet Module, see "Uninstalling RosettaNet Module 7.1 SP2" on page 35.

   Important: Go to *Integration Server_directory*\packages directory and remove the following RosettaNet Module related packages:

   - WmRosettaNet
   - WmRNIF11TRP (if present)
   - WmRNIF20TRP (if present)

3. Go to *Integration Server_directory* \updates directory, and delete all the RosettaNet Module jars. An example for the name of a RosettaNet Module jar file is RN_7.1_SP1_FixXX_XX.jar.
4. Install RosettaNet Module 7.1 SP2 on a supported version of Integration Server. For installation instructions, see "Installing RosettaNet Module 7.1 SP2" on page 30. For a list of supported Integration Server versions, see *webMethods eStandards Modules System Requirements* .

5. Download and install the latest fixes for RosettaNet Module and any other webMethods products you installed from the Empower Product Support website at https://empower.softwareag.com. For information about installing the fixes, see instructions in the corresponding fix readme file.
6. If you have configured RosettaNet Module cache clearing settings in *Integration Server_directory* \config\Caching\webm-cache-config.xml file, verify that the name of the RosettaNet Module cache is `RNModelSessionCache`. If a different name is shown for the cache, change it to `RNModelSessionCache`.
7. If required, open the config.cfg file from *Integration Server_directory* \packages \WmRosettaNet\config directory, and edit the required properties to reflect the values in your backed up config.cfg file.
8. Verify that the upgrade was successful by initiating a RosettaNet business process model and ensuring that RosettaNet transactions have completed successfully.

Upgrading from RosettaNet Module 6.0.1

Follow these steps to upgrade and migrate from RosettaNet Module 6.0.1 (and its related service packs) to RosettaNet Module 7.1 SP2.

To migrate...	Use...
Document types, trading partner agreements, and partner profiles	Trading Networks migration scripts.
Business process models	The "pub.estd.rosettaNet:migrate" on page 98 service.
Packages, folders, and flow services	The "pub.estd.rosettaNet:migrate" on page 98 service and edit custom services.

To migrate from RosettaNet Module 6.0.1 to RosettaNet Module 7.1 SP2

1. Back up the entire existing webMethods installation directory on each machine, and back up your webMethods databases as instructed by your RDBMS vendor.
2. Install RosettaNet Module 7.1 SP2 on a supported version of Integration Server. For installation instructions, see "Installing RosettaNet Module 7.1 SP2" on page 30. For a list of supported Integration Server versions, see *webMethods eStandards Modules System Requirements* .
3. Migrate Trading Networks document types, trading partner agreements, and partner profiles as described in the webMethods upgrade guide for your release. See “About this Guide” for specific document titles.

Important: Do not migrate the PIP 0A1 related TPAs because in RosettaNet Module 7.1 SP2, NOF is handled within the PIP process model and does not require a separate process model.

4. Migrate RosettaNet Module 6.0.1 packages, folders, flow services, and process models as follows:

   a. In Designer, import the RosettaNet business process models from RosettaNet Module 6.0.1 to RosettaNet Module 7.1 SP2 using a supported version of Designer (as noted in *webMethods eStandards Modules System Requirements*).

   b. Copy the following to *Integration Server_directory* \packages directory.

      - The custom package that contains reference of WmIPRoot services.
      - The generated package of 6.0.1 process models.

      In RosettaNet Module 6.0.1, Notification of Failure (NOF) documents were handled in a separate NOF (PIP 0A1) process model. In RosettaNet Module 7.1 SP2, NOF is handled within the PIP process model and does not require a separate process model. Due to this change, you must not migrate PIP 0A1 related TPAs, IS documents, TN XML documents types, and process models.

   c. Build and upload the process models.

   d. Use the "pub.estd.rosettaNet:migrate" on page 98 service to migrate the services from the package, generated when uploading the process model. In the service's *listOfComponents* input parameter, enter the package name. Similarly you can migrate your custom packages and services.

   e. Edit any custom services that reference the WmRNIF11TRP and WmRNIF20TRP packages to now point to the WmRosettaNet public folder. For example, change wm.ip.rnif11.util:getRNIF11Documents and wm.ip.rnif20.util:getRNIF20Documents to pub.estd.rosettaNet:getRNOMimePart.

      Note: See the nsMap.properties file, located in the WmRosettaNet package \migrate folder for a complete list of services that must be remapped to services in the WmRosettaNet and WmEstdCommonLib packages. Public services that were available in the WmRNIF20TRP and WmRNIF11TRP packages in RosettaNet Module 6.0.1 have been consolidated and moved to the WmRosettaNet package in RosettaNet Module 7.1 SP2. The WmRosettaNet package automatically selects the appropriate protocol-related details for RNIF 1.1 or RNIF 2.0, as governed by the TPA settings.

      In RosettaNet Module 6.0.1, focal roles were set in the process models. As of RosettaNet Module 7.1, focal roles are defined in the Trading Partner Agreement associated with the process, as *ProcessInfo/SenderFocalRole* and P*rocessInfo/ ReceiverFocalRole* , for the sender and receiver, respectively.

   f. Restart the Integration Server.

5. In Designer, verify that the upgrade was successful by initiating the RosettaNet business process models. Verify that RosettaNet transactions complete successfully.

Uninstalling RosettaNet Module 7.1 SP2

The instructions in this section are meant to be used with the uninstallation instructions in the webMethods installation guide.

To uninstall RosettaNet Module 7.1 SP2

1. Shut down the Integration Server that hosts RosettaNet Module 7.1 SP2.
2. Start the Software AG Uninstaller, selecting the webMethods installation directory that contains the host Integration Server.
3. In the product selection list, select **eStandards > webMethods RosettaNet Module 7.1 SP2 > Program Files**. Also select the transport versions used (RNIF 1.1 or RNIF 2.0), and any other products and items you want to uninstall.
4. Restart the host Integration Server.

   Software AG Uninstaller removes all RosettaNet Module 7.1 SP2-related files installed into the *Integration Server_directory*\packages directory. However, it does not delete files created after you installed the module (for example, user-created, configuration, or archive files), nor does it delete the module directory structure. You can navigate to the *Integration Server_directory*\packages directory and delete the RosettaNet specific folders manually.

3 Getting Started: Steps for Implementing a PIP

Implementing a RosettaNet PIP

The following procedure outlines the steps for implementing a RosettaNet PIP.

Important: The following procedure assumes that you have already installed webMethods Integration Server, webMethods Trading Networks, Software AG Designer, My webMethods Server, and webMethods RosettaNet Module.

Step 1: Define Trading Partner Profiles

Using My webMethods, define the profiles for your enterprise 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 "Defining Trading Partner Profiles " on page 41.
For detailed information about Trading Networks assets, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Step 2: Import relevant PIP Specification or PIP Archive

To import PIP specification file, download the required PIP specification file from the RosettaNet Website (http://www.rosettanet.org) and import the downloaded PIP using PIPTools. For more information about importing the PIP using PIPTools, see *webMethods PIP Tools Installation and User's Guide*.

To import PIP Archive, download the compressed PIP Archives, also called Process Archives, that represent the version of the PIPs that you want to implement. Install these archives on Integration Server using the import functionality in RosettaNet Module. For more information on importing PIP Archives, see "Importing PIP Archives" on page 51.

Step 3: Define TN XML Document Types

TN XML document types are included in the PIP archives or PIP Specification zip file that you import into Integration Server. When you import a PIP archive or PIP Specification zip file, the TN XML document types for the respective PIPs are automatically created in Trading Networks. However, you may want to create your own *internal* TN XML document types or modify the provided internal TN XML document types. For more information about defining TN XML document types, see "Defining Internal TN XML Document Types" on page 57.

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 4: Customize your Trading Partner Agreements

A trading partner agreement (TPA) is a set of parameters that you can use to govern how documents are exchanged between two trading partners. The required set of TPAs for a PIP will be automatically created during the PIP import process. Using My webMethods, customize your TPAs by specifying the sender and receiver and, for an initiator's TPA, 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 using TPAs in RosettaNet Module, see "Customizing Trading Partner Agreements" on page 61. For detailed information about Trading Networks assets, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Step 5: Write Inbound and Outbound Mapping Services

There are two types of mapping services: inbound and outbound. Create an inbound mapping service to map a RosettaNet PIP document received from a trading partner into the format used by your internal (back-end) system. Create an outbound mapping service to map a business document generated by your internal system into a RosettaNet PIP document that you can send to a trading partner. For more information on mapping business documents, see "Mapping a RosettaNet PIP Document" on page 71.

Step 6: Write Error Handler Services

Write error handler services, if necessary. 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 "Error and Exception Handling" on page 107.

Step 7: Customize Process Model Templates

Determine which process model template you need and customize it according to the PIP you are implementing. For example, you might customize a process model template to specify the TN XML document type for a particular wait step and assign inbound and outbound mapping services to the relevant mapping steps. For more information about customizing process model templates, see "Customizing Process Model Template" on page 79. For more information about process models in general, see the Designer online help.

Step 8: Build and Upload Your Process Model

Build and upload your process model in Designer. For information about building and uploading your process model, see the Designer online help.

Step 9: Start and Run a Conversation

To start a conversation on the initiator's side, the initiator's internal system invokes the "pub.estd.rosettaNet:processDocument" on page 99 service. 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 the "pub.estd.rosettaNet:receive" on page 100 service. After you start a conversation (or process), you can monitor its progress in Monitor.

For more information about starting a conversation, see "RosettaNet Conversation Flow" on page 88. For more information about monitoring conversations, see "About webMethods Monitor " on page 91 and *webMethods Monitor User's Guide*.

4 Defining Trading Partner Profiles

What Is a Trading Partner?

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

In addition to creating trading partner profiles for all of your trading partners, you must also create a profile for your own organization.

Why Are Trading Partner Profiles Important?

Your trading partner profiles, used along 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) and fulfiller (seller), in a process model define the actions you perform for a transaction type, as well as the actions you expect your trading partners to perform during those transactions. In short, the configuration of process models, the application of TPAs, and the implementation of PIPs are what enable you to successfully interact with your trading partners.

You may need to perform different roles for different trading partners. For example, you might need to act as a buyer toward your suppliers and as a seller toward your distributors. In RosettaNet Module, you can define your trading partners, using trading partner profiles, and the roles the trading partners perform in transactions, using process models for each role:

- For your organization, define your trading partner profile, known as the *enterprise* profile. Then, customize a process model template for each role that you perform with your trading partners. If you perform multiple roles, you must have a separate process model for each role.
- For your trading partners, define *one* trading partner profile for each trading partner with whom you conduct transactions. Next, customize a process model template for each role the partner performs. Again, if the partner performs multiple roles, there must be a separate process model for each role.

A trading partner's role determines how you customize the process model template. When you define the process model, specify the sender and receiver of the business document by assigning a trading partner profile to each role. For information about customizing process model templates, see "Customizing Process Model Template" on page 79. For information about process models in general, see the Designer online help for your release. See “About this Guide” for specific document titles.

Defining Your Enterprise Profile

Before exchanging business documents with your trading partners, define trading partner profiles in My webMethods. You must first define the trading partner profile for your organization, known as the *Enterprise* profile.

To complete your enterprise profile definition, do the following tasks:

- Define the required profile fields.
- Define contact information.
- Configure the document delivery method.
- Define the security certificate.
- Activate the profile.

These procedures only describe the required RosettaNet-specific information. For complete procedural information on defining your enterprise profile and to find field descriptions, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Step 1: Defining Profile Fields

This procedure describes the required profile fields in the enterprise profile. For complete details about the remaining fields in the partner profile, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

To define enterprise profile fields

1. In My webMethods, select **Administration > Integration > B2B > Partner Administration > Partner Profiles** to open the Partner Profiles page.
2. Define the following required fields in your enterprise profile:
   a. In the **Corporation Name** field, type the name of your organization.
   b. Select the **External IDs** tab and click **Add ID**.
   c. Add a `DUNS` **ID Type** and define the value as your enterprise's D-U-N-S number.
3. Click **Save**.

Step 2: Defining Contact Information

To use RNIF version 2.0 as your transport protocol, you should define at least one contact in your enterprise profile, including a valid e-mail address and phone number. RosettaNet Module uses this information to process a PIP 0A1 Notification of Failure when there is a communication failure with your trading partners.

Note: In RNIF version 2.0 delivery header, the value of *locationID* is populated from the **Address** field of the profile. Therefore, to populate locationID, you must update your address in your enterprise profile.

To define contacts, follow the instructions in "Managing the Contacts in Your Profile" in the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Step 3: Configuring Delivery Settings

You may define more than one delivery method for sending documents to your partners, but you must specify *at least one* delivery method as the *preferred delivery method* in the partner profile. If no delivery method is defined, RosettaNet Module cannot exchange documents or execute any conversations with your partners.

To define the preferred delivery method, follow the instructions in "Managing Document Delivery Methods in a Enterprise Profile" in the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

If you are using HTTP as the delivery method, you must specify the RosettaNet receive service, "pub.estd.rosettaNet:receive" on page 100, as the **Location**, as explained in the following procedure.

To define HTTP as the preferred delivery method

1. In My webMethods, select **Administration > Integration > B2B > Partner Administration > Partner Profiles** to open the Partner Profiles page.
2. Open your enterprise partner profile and select the **Delivery Settings** tab.
3. In the Delivery Methods section of the screen, click **Add Delivery Method**.
4. In the **Delivery Method** list, select Primary HTTP or Secondary HTTP as the delivery method.
   a. In the **Location** field, specify the value as /invoke/pub.estd.rosettaNet/receive. For backward compatibility, you can specify the deprecated location, /invoke/wm.ip.rn/receive.
   b. Complete the remaining fields as specified in the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Step 4: Defining Security Information

RosettaNet Module uses the same certificate as Trading Networks for signing and encrypting messages.

In My webMethods, on the Partner Profiles page, select the **Certificates** tab and add a new certificate by following the instructions in the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Step 5: Activating Your Enterprise Profile

You must activate (or enable) your enterprise profile before you can exchange documents with trading partners. For instructions on enabling your profile, see “Enabling Your Profile-Changing the Status to Active” in the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

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, define your trading partners' profiles in My webMethods.

To complete your partner profile definition, do the following tasks:

- Define the required profile fields.
- Define contact information.
- Configure the document delivery method.
- Define the security certificate.
- Activate the profile.

This section describes the required RosettaNet-specific partner profile information. For complete procedural information on defining your partners’ profiles and to find field descriptions, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Step 1: Defining Profile Fields

This procedure describes the required fields in the partner profile. For complete details about the remaining fields in the partner profile, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

To define partner profile fields

1. In My webMethods, select **Administration > Integration > B2B > Partner Administration > Partner Profiles** to open the Partner Profiles page.
2. Define the following required fields in your partner’s profile:
   a. In the **Corporation Name** field, type the name of your partner.

b. Select the **External IDs** tab and click **Add ID**.

c. Add a DUNS **ID Type** and define the value as your trading partner's D-U-N-S number.

3. Click **Save**.

Step 2: Defining Contact Information

To use RNIF version 2.0 as the transport protocol with your trading partner, you should define at least one contact in your trading partner profile, including a valid e-mail address and phone number.

Note: In RNIF version 2.0 delivery header, the value of *locationID* is populated from the **Address** field of the trading partner profile. Therefore, if you want locationID populated, you must update the address in the relevant partner's profile.

To define contacts, follow the instructions in "Managing the Contacts in Your Profile" in the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Step 3: Configuring Delivery Settings

You may define more than one delivery method for sending documents to your partners, but you must specify *at least one* delivery method as the *preferred delivery method* in the partner profile. If no delivery method is defined, RosettaNet Module cannot exchange documents or execute any conversations with your partners.

To define the preferred delivery method, follow the instructions in "Managing Document Delivery Methods in a Partner Profile" in the Trading Networks administration guide for your release. See "About this Guide" for specific document titles..

If you are using HTTP as the delivery method, you must specify the RosettaNet receive service, "pub.estd.rosettaNet:receive" on page 100, as the **Location**, as explained in the following procedure.

To define HTTP as the preferred delivery method

1. In My webMethods, select **Administration > Integration > B2B > Partner Administration > Partner Profiles** to open the Partner Profiles page.
2. Open your partner's trading partner profile and select the **Delivery Settings** tab.
3. In the Delivery Methods section of the screen, click **Add Delivery Method**.
4. In the **Delivery Method** list, select Primary HTTP or Secondary HTTP as the delivery method. Trading Networks displays the fields related to the selected method.

   a. Check the option to set this delivery method as the preferred delivery method.

b. In the **Location** field, specify the value as /invoke/pub.estd.rosettaNet/receive. For backward compatibility, you can specify the deprecated location, /invoke/wm.ip.rn/receive.

c. Complete the remaining fields as specified in the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Step 4: Defining Security Information

RosettaNet Module uses the same certificate as Trading Networks for signing and decrypting messages.

In My webMethods, on the Partner Profiles page, select the **Certificates** tab and add a new certificate by following the instructions in the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Step 5: Activating Your Trading Partner's Profile

You must activate (enable) your trading partner's profile before you can exchange documents with the partner. For instructions on enabling the partner profile, see "Enabling A Partner's Profile-Changing the Status to Active" in the Trading Networks administration guide. See "About this Guide" for specific document titles.

5 Importing PIP Specification/PIP Archives

What is a PIP Specification?

A *PIP Specification file* consists of the PIP, available from the RosettaNet Website. More information about type of PIPs, contents of specification file, and their organization is also available at the RosettaNet website (http://www.rosettanet.org). For details on importing the PIP Specification file, see *webMethods PIP Tools Installation and User's Guide*.

What is a PIP Archive?

A *PIP archive* is a compressed file (.par), also called a *process archive*, with the 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.

What Does a PIP Archive Contain?

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

- **IS document types**— Defines the namespace for the PIP and the structure of the RosettaNet PIP documents that are exchanged during the PIP conversation.
- **TN XML document types (internal and external)**—Used by Trading Networks to identify and route PIP documents.
- **Trading Partner Agreements (TPAs)**—Includes one for the initiator of a conversation and one for the fulfiller of a conversation.

Why Do I Need to Import a PIP Specification / PIP Archive?

To implement a PIP in RosettaNet Module, you must import a PIP Specification file or a PIP archive. You can implement different versions of the same PIP to support different transactions between you and your trading partners. You can also modify the standard PIPs provided by RosettaNet and share them with your trading partners. However, in any specific PIP conversation, you and your trading partner must be using the same version of the PIP.

PIPs and Process Models

Each RosettaNet PIP implements a business task or process. The differences between the specific PIP processes are mostly in the business documents that are exchanged and how those business documents are handled.

"Concepts" on page 15, provides a basic overview of PIP transactions using the example of the PIP 3A4 Manage Purchase Order process, 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. 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 "Customizing Process Model Template" on page 79.
- **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. 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 "Customizing Process Model Template" on page 79.
- **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 Specification File / PIP Archives

You can import a PIP Specification or a PIP Archive as explained below.

Importing PIP Specification

For importing PIP Specification file, see *webMethods PIP Tools Installation and User's Guide*.

Importing PIP Archives

You can import a PIP archive by using the RosettaNet import facility in Integration Server Administrator.

Note: You must import PIP 0A1 Notification of Failure because this PIP is used by most other PIPs to communicate errors to your trading partners. This is the SamplePIP0A1.par, available on the Communities Website (http://communities.softwareag.com/ecosystem/communities/public/Developer/webmethods/codesamples/RosettaNet_801version.html).

To import PIP archives into Integration Server

1. Save the file you want to import, to the import subdirectory of RosettaNet Module:

   Integration Server_directory\packages\WmRosettaNet\import

2. Change the extension of the imported PIP file to *.par*. RosettaNet Module only lists .par files.
3. After saving the file to your file system, start Integration Server and Integration Server Administrator.

   Note: For detailed information about using Integration Server, see the Integration Server administration guide for your release. See "About this Guide" for specific document titles.

4. In Integration Server Administrator, select **Adapters > RosettaNet**.
5. Click **PIP Import**.
6. On the **PIP Import Files** page, select the check box next to the PIP you want to import.
7. Click **View**.
8. If there is an exclamation point icon to the left of the entries in the **Archive Contents** section of the **PIP Import File Details** table, you have already installed this version of this PIP. If you want to use the version in the archive you are importing, continue to next step. Before importing, if you want to save your custom changes in the existing PIP IS document types, follow the procedure in "Exporting PIP Archives" on page 54.
9. Scroll to the bottom of the page and click **Import Archives**.

Customizing PIPs

When implementing your business solution using RosettaNet Module, you can modify 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 "Customizing and Exporting PIP Specification / PIP Archives: Advantages and Disadvantages" on page 55.

Customizing IS Document Types for Existing Standard RosettaNet PIPs

You can customize an IS document type for an existing standard RosettaNet PIP. You may want to do this, for example, if the PIP has fields that your trading partners are unable to populate, or are not needed for your business needs. There are two options for altering an IS document type:

- Change unnecessary fields to make them optional.

- **Advantage:** Allows documents for that PIP to pass validation.
- **Disadvantage:** Affects every trading partner exchanging this PIP with you.

- Isolate changes to the IS document type for a specific trading partner only.
  - **Advantage:** Localizes changes for one trading partner and one particular PIP only.
  - **Disadvantage:** Creates an additional IS document type to maintain.

To isolate changes for an IS document type, follow these steps:

1. Create a folder in the WmRNPips package under wm.b2b.rn.rec.PIPs and name it <*your customized PIP structure*>.
2. Enter the new location as the value for the *NSFolder* parameter in the TPA for that PIP.

Implementing a Custom PIP

RosettaNet Module allows you to create and implement custom PIPs. Complete the following procedure to create a custom PIP.

To implement a custom PIP

1. In My webMethods, select **Administration > Integration > B2B** to create a new TN XML document type as follows:

   Note: For help with this step, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

   a. Specify a root tag that matches the root tag of the customized PIP.

   b. Add a pipeline variable matching the new TN XML document type called *processVersion* . Define *processVersion* as the version of the PIP you are implementing.

2. Create a new IS document type for the customized PIP using Designer.

   Note: For more information about process models and Software AG Designer, see the Designer online help for your release. See "About this Guide" for specific document titles.

3. In My webMethods, 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.
4. In Software AG Designer, in the process model template for the PIP you are implementing, specify the new TN XML document type as input to the appropriate receive step.

Exporting PIP Archives

You can create a custom PIP by changing the standard PIP provided by RosettaNet. Note, however, that if you do not save the customized PIP with a different name than the standard, you risk overwriting your customized PIP the next time RosettaNet releases a new version of the standard PIP.

To avoid losing customized PIPs whenever RosettaNet releases new standards, complete the following procedure to export the customized PIP.

To export a PIP archive from Integration Server

1. In Integration Server Administrator, select **Adapters > RosettaNet**.
2. On the RosettaNet Management page, select **PIP Export** to display the Create PIP Export Archive page.
3. Define the PIP archive properties as follows:
   a. In **File Name**, define the file name of the PIP archive, for example, PIP3A4v1_1.par for PIP that was customized from PIP 3A4.
   b. In **PIP Name**, define the name of the PIP, for example, Manage Purchase Order Custom for a PIP customized from PIP 3A4 Manage Purchase Order.
   c. In **PIP Number**, specify the PIP number, for example, 3A4.
   d. In **PIP Version**, specify the PIP version, for example, 1.1.

   Note: The version numbering of the custom PIP should mirror that of the standard PIP. However, subsequent updates to the customized PIP may require diverging from standard PIP version number.
4. Define the Trading Networks properties of the PIP, as follows:
   a. In the **Select TN Biz Doc Types to Export** section, select the check boxes for each of the TN XML document types that you want to include as part of the PIP archive file.

   Note: Select all of the TN XML document types that were included with the standard version of the PIP on which you based this customized version, as well as any new TN XML document types you created for the custom PIP.

   b. In the **Select TPAs to Export** section, select the check boxes for the Trading Partner Agreements (TPAs) that you want to include in the PIP archive file.
5. In the **Select PIP Folders to Export** section, select each of the PIP Integration Server folders that you want to include in the PIP archive file.

> **Note:** Select only the PIP Integration Server folders that were included with the standard version of the PIP on which you based this customized version.

6. Click **Create Archive**.

RosettaNet Module generates a PIP archive file (.par) in the folder *Integration Server_directory*\packages\WmRosettaNet\export.

Customizing and Exporting PIP Specification / PIP Archives: Advantages and Disadvantages

The following are the advantages and disadvantages of customizing PIPs and PIP archives.

Advantages

- Enables RosettaNet Module to manage your custom PIP as distinct from the standard PIP on which you based your customization.
- Enables you to send the custom PIP to your trading partners (if they are also using RosettaNet Module).
- Enables you to do your own version management on your customized PIPs.

Disadvantages

- After you create a custom PIP, you must manually keep the custom process in parallel with the standard process from that point on (for example, when new features are added to the process).
- 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, TPAs 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 wm.b2b.rn.rec.PIPs folder).

  To save the services, you must create your own package on 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 Designer.

6 Defining Internal TN XML Document Types

What are TN XML Document Types?

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

When RosettaNet Module receives a business document, it invokes a Trading Networks service to recognize the type of business document. For more information about TN XML document types, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

TN XML Document Types Provided with webMethods RosettaNet Module

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

TN XML document type	RNIF Version	Description
Acceptance Acknowledgement	1.1	Document type for an acceptance acknowledgment.
AcceptanceAcknowledgement Exception	1.1	Document type for an exception for an acceptance acknowledgment.
Exception (RNIF 1.1)	1.1	Document type for a RNIF 1.1 exception.
Exception (RNIF 2.0)	2.0	Document type for a RNIF 2.0 exception.
ReceiptAcknowledgement (RNIF 1.1)	1.1	Document type for a receipt acknowledgment.
ReceiptAcknowledgementException	1.1	Document type for an exception for a receipt acknowledgment.
ReceiptAcknowledgment (RNIF 2.0)	2.0	Document type for a receipt acknowledgment.
RosettaNet Dummy Doc Type		Dummy document type used in the RosettaNet process model templates.

TN XML document type	RNIF Version	Description
WmRNTestServiceDocument Test Service		Dummy document used by the RosettaNet Module ping service.
RN11BizDocType	1.1	Document type Placeholder for failed document processing.
RN20BizDocType	2.0	Document type Placeholder for failed document processing.

You can view these TN XML document types in My webMethods.

Defining Your Own Internal TN XML Document Types

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

Important: Do not modify any of the provided *external* TN XML document types. If you do, Trading Networks will not be able to join an incoming business document with a conversation.

When you define an internal TN XML document type, specify the following:

- The root tag from the business document that must match the TN XML document type.
- The XML queries that Trading Networks uses to extract the *SenderID* and *ReceiverID* attributes from the business document.

Trading Networks uses XML queries to extract the *SenderID* and *ReceiverID* attributes from business documents that are sent to RosettaNet Module from an internal system. The *SenderID* and *ReceiverID* indicate the sender and receiver of the business document, respectively.

For more information on defining TN XML document types, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Note: You can determine if a TN XML document type is external by the description and the pipeline. If the document type is external, it will always have a pipeline matching variable of *processVersion* . Internal TN XML document types do not have a matching variable in the pipeline.

7 Customizing Trading Partner Agreements

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) or PIP Specification file into Integration Server, the TPAs for that PIP are created in Trading Networks. You can view and customize the TPAs in My webMethods. For information about working with TPAs, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Every PIP in RosettaNet Module is associated with two TPAs, one for the sender in a conversation and one for the receiver. The TPAs 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 to which the parameters map vary between RNIF version 1.1 and 2.0.

How Does RosettaNet Module Identify a TPA?

Every TPA is uniquely identified by a sender, a receiver, and an agreement ID. During a conversation between trading partners, RosettaNet Module uses this information to retrieve the TPAs for the sender and receiver 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 *sender* of a PIP conversation, the first TN XML document type name is typically the TN XML document type of the document generated by the internal (back-end) system. For example, in the PIP 3A4 Request Purchase Order example described in "Processing Overview" on page 20, the TPA agreement ID for the “Buyer” conversation might be “Internal PO Request.”
- For the *receiver* 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 Request Purchase Order example described in "Processing Overview" on page 20, the TPA agreement ID for the “Seller” conversation might be “PIP3A4 v1.1 Purchase Order Request Action Document.”

Modifying the TPA

This section describes how to modify a TPA to define the sender or receiver of the RosettaNet PIP business document.

Use My webMethods to view and modify the details of a TPA. My webMethods initially displays the **Sender** and **Receiver** fields of the TPA with the default value of "Unknown."

To modify the TPA

1. In My webMethods, define both the **Sender** and **Receiver** as a specific organization, or define both as "Unknown." For further information about working with TPAs, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles. Refer to the table below for additional details about the TPA fields.

   Important: The TPA must specify a value for both the **Sender** and **Receiver** (for example, *Company1* and *Company2* , respectively) or define both **Sender** and **Receiver** as *Unknown* . You must not define a TPA where one value is known and the other is *Unknown* , for example, where **Sender** is *Company1* and the **Receiver** is *Unknown* .

2. If you are defining the initiator's TPA only, define the **Agreement ID** as the TN XML document type for the document generated by your internal system.

   Important: If you are defining the receiver's TPA, do not modify the **Agreement ID**. Modifying this value for the receiver prevents RosettaNet Module from being able to identify the TN XML document type for the business document.

3. Define the remaining fields of the TPA as indicated:

TPA Field	Description
Sender	The name of your organization's profile.
Receiver	The name of your trading partner's profile.
Agreement ID	An application-specific field that uniquely identifies the type of agreement between two partners.
IS Document Type	The "wm.ip.rn.rec:UserParameters IS Document Type" on page 64, included in the WmRosettaNet package, specifies the data or parameters that you define in the TPA.
Data Status	When the **Agreement Status** is `Agreed`, this field indicates whether you can update the values for the TPA parameters in the IS document type. The data status can be one of the following: ■ Modifiable—TPA data can be changed.

TPA Field	Description
	■ Non-modifiable—TPA data cannot be changed.

For more detailed instructions on modifying a TPA, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Distinguishing between the Sender’s and Receiver’s TPA

RosettaNet Module distinguishes between the sender’s and receiver’s TPAs by the agreement ID. The agreement ID for the receiver's TPA always includes the PIP name, for example, PIP3A4 v1.1 Purchase Order Request Action Document.

You may want to delete the TPAs that you do not use. For example, if you are only always the sender in a conversation, you can choose to delete the receiver's TPA.

For a list and description of the TPA parameters, see the next section, "Parameter Settings" on page 64.

Parameter Settings

TPA parameters are divided into three sections:

- **ProcessInfo.** Consists of process-level information such as the Transport and TransportVersion.
- **PipInfo.** Consists of information about the PIP process, such as the DTD and ActionCode. 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 and EncryptPayload.

The following section "wm.ip.rn.rec:UserParameters IS Document Type" on page 64 describes the TPA parameters for an RNIF 1.1 TPA and an RNIF 2.0 TPA.

wm.ip.rn.rec:UserParameters IS Document Type

The wm.ip.rn.rec:UserParameters IS document type describes the user parameters used in conversations for all PIPs. This structure is used internally and is not meant for general modification.

The TPA parameters described below, are the parameters part of the wm.ip.rn.rec:UserParameters IS document type.

TPA Section	Parameter	Description
ProcessInfo		
	Transport	Transport protocol type (for example, RNIF).
	TransportVersion	Version of the transport protocol (1.1 or 2.0).
	SenderClassification	Business classification of the sender (for example, manufacturer, distributor, or customs broker). This is populated in the GlobalPartnerClassificationCode tag describing the *fromPartner* in the service header. For information about the valid values, see the *fromPartner* element details in the PIP specification corresponding to the executing PIP, activity, or action.
	ReceiverClassification	Business classification of the receiver (for example, buyer). This is populated in the GlobalPartnerClassificationCode tag describing the *toPartner* in the service header. If the current message is a signal, the value of the *fromPartner* in the signal must be the same as that of the *toPartner* in the Action to which this signal is replying. For information about the valid values, see the *toPartner* element details in the PIP specification corresponding to the executing PIP, activity, or action.
	Sign	Whether the outbound RosettaNet PIP document for the conversation should be signed. Valid values are `Yes` or `No`.
	SignatureRequired	Indicates if the inbound PIP document should contain your trading partner's signature. Valid values are `Yes` or `No`.

TPA Section	Parameter	Description
	HashAlgorithm	Algorithm used to generate the RosettaNet Message Digest (MD5 or SHA-1). **Note:** MD5 is not supported for RNIF version 2.0. Even if MD5 is selected for RNIF version 2.0, SHA-1 will be used.
	ValidationService	Service that validates the RosettaNet PIP document. If you want to use custom service for inbound validation of payload, provide the path of custom service. To bypass inbound validation of a document, provide a dummy service. If left blank, validation of the payload is performed against the PIP document. For large documents, the user should define the validation service to use. **Note:** If your process model handles both large and small business documents, your validation service also must handle both types of documents.
	ValidateOutput	Indicates whether to validate the PIP document before sending it to the trading partner. Valid values are `Yes` or `No`.
	SenderFocalRole	This is populated in the GlobalPartnerRoleClassificationCode tag describing *fromRole* in the service header. Additionally, this is also populated in the GlobalBusinessServiceCode tag describing *fromService* in the service header. For valid values, see the *fromService* and *fromRole* details in the PIP specification corresponding to the executing PIP, activity, or action.
	ReceiverFocalRole	This is populated in the GlobalPartnerRoleClassificationCode tag describing *toRole* in the service header.

TPA Section	Parameter	Description
		Additionally, this is also populated in the GlobalBusinessServiceCode tag describing *toService* in the service header. If the current message is a signal, the value of *fromService* must be the same as *toService* in the action to which this signal is replying. For valid values, see the *toRole* and *toService* details in the PIP specification corresponding to executing PIP, activity, or action. **Note:** Focal roles must be set correctly before running a transaction or the transaction will fail. The sender and receiver focal roles are reversed in the TPAs of the two trading partners exchanging documents for a two action PIP. For example, the focal role in the TPAs for Partner A, acting in role X, and Partner B, acting in role Y, would be as follows: ■ Partner A TPA: ■ SenderFocalRole = X ■ ReceiverFocalRole = Y ■ Partner B TPA: ■ SenderFocalRole = Y ■ ReceiverFocalRole = X Focal roles are an important part of the *ConversationID* generated by RosettaNet Module. The *ConversationID* is generated internally, in the format: `Initiator DUNS-%%-UniqueID-%%-SenderFocalRole` (where "-%%-" is used as the separator).
PIPInfo		
	ProcessCode	Code or name of the PIP, such as 3A4.

TPA Section	Parameter	Description
	ProcessVersion	Version of the PIP process, such as 1.1 or 1.4.
	ProcessDescription	Description of the PIP process (for example, the description of PIP3A4 is "Request Purchase Order").
	TransactionCode	Transaction associated with each RosettaNet PIP document.
	ActionCode	Action associated with each RosettaNet PIP document.
	DTD	File name of the Data Type Definition for the RosettaNet PIP document.
	NSDecls	XML namespace declaration with the name and value that will be added to the XML payload.
	NSFolder	Folder that contains the IS document types against which the RosettaNet PIP documents are validated. **Note:** This folder should be located in the directory, *Integration Server_directory*\packages\WmRNPips\ns\wm\b2b\rn\rec\PIPs. If NSFolder is left blank, webMethods RosettaNet Module will search in the *Integration Server_directory*\packages\WmRNPips\ns directory for the IS document types.
	SchemaValidation	Whether schema validation for the documents is required. Valid values are `Yes` or `No`. If you are using schema based PIP, you must set the value as `Yes`, and also set the value for the *ISSchema* parameter. If you do not set these two values, the inbound

TPA Section	Parameter	Description
		validation for the schema based PIP will fail. To bypass the inbound validation, provide a dummy service as value for the *ValidationService* parameter that is part of **ProcessInfo** TPA Section.
	ISSchema	Name of the XML schema against which the document should be validated.
	UsageCode	Indicates if the RosettaNet conversation is in test or production mode. Valid values are `Test` or `Production`.
RNIF2.0		
	Encoding	Specifies how to encode MIME parts. **Note:** Use `base64` and `uuencode` encoding for binary content messages.
	EncryptPayload	Whether to encrypt the outbound RosettaNet payload. Valid values are `Yes` or `No`.
	EncryptHeader	Whether to encrypt the service header of the RosettaNet PIP document. Valid values are `Yes` or `No`. **Note:** You cannot encrypt the service header alone, without encrypting the payload. To encrypt the payload and service header, set the value of *EncryptPayload* and *EncryptHeader* parameters as `Yes`.
	EncryptionAlgorithm	Algorithm used to encrypt the outbound RosettaNet PIP document. Valid values are `Triple DES`, `RC2`, and `DES`.
	RC2EncryptionKeyLength	Length (in bits) of the RC2 encryption key. Valid values are `128`, `64`, and `40`.

TPA Section	Parameter	Description
	ThirdPartyPayloadType	Payload type for the third-party specification. Only the third party Open Applications Group (OAG) is supported. This parameter is equivalent to the *ProcessCode* parameter for PIPs. **Note:** If you are using RosettaNet for PIDX, you must update this *ThirdPartyPayloadType* parameter.
	ThirdPartyPayloadVersion	Payload version for the third-party specification. Only the third party Open Applications Group (OAG) is supported. This parameter is equivalent to the *ProcessVersion* parameter for PIPs. **Note:** If you are using RosettaNet for PIDX, you must update this *ThirdPartyPayloadVersion parameter* .
	ThirdPartyReferenceId	Payload reference ID for the third-party specification. Only the third party Open Applications Group (OAG) is supported. **Note:** If you are using RosettaNet for PIDX, you must update this *ThirdPartyReferenceId* parameter.
	CompressContent	Whether to compress the outbound document. Valid values are `Yes` or `No`.
	sync	Whether to send response or receipt acknowledgement synchronously. Valid values are `Yes` or `No`. **Note:** If you are using a synchronous PIP, for example, PIP 2A9, set the value of *sync* parameter as `Yes`.

8 Mapping a RosettaNet PIP Document

What is Business Document “Mapping?”

Mapping is the process of relating the structure, values, or content of a business document in one format to the values and fields in an alternate business document format. You need to create maps between business documents because, for example, your internal (back-end) system may render data in a different format than that of the RosettaNet PIP documents. Transforming a document generated by your back-end system into RosettaNet format allows you to exchange data with your partners in a common format.

Note: For information about mapping large documents, see "Processing Large Business Documents" on page 123.

Why Create an Outbound Mapping Service?

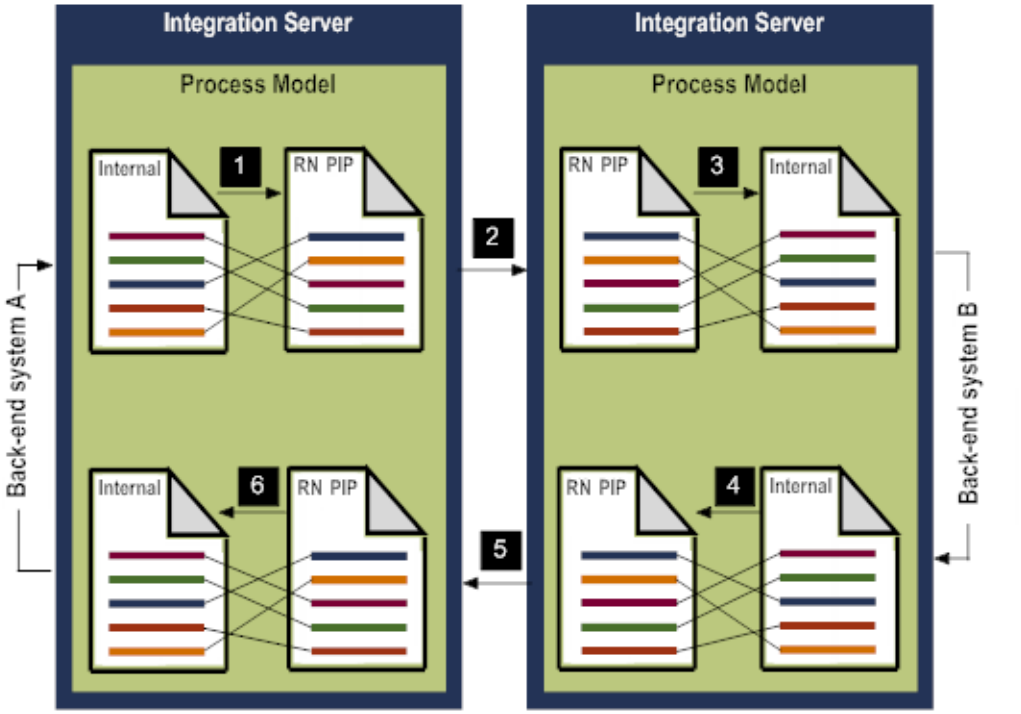
Create an outbound mapping service to translate a document generated by your internal system in proprietary format (for example, an IDOC from an SAP R/3 system) into a RosettaNet PIP document that you can send to your trading partners. Elements of the proprietary business document must be mapped to corresponding elements in a specific version of a RosettaNet PIP (for example, PIP 3A4 Request Purchase Order, version 1.1) document.

Why Create an Inbound Mapping Service?

Create an inbound mapping service to map each element of the inbound RosettaNet PIP document to a corresponding data element in your internal system’s proprietary format. For example, if you use SAP R/3 as your internal ERP system, and you want to receive a RosettaNet PIP document from RosettaNet Module, you must create an inbound mapping service that maps each element of the RosettaNet PIP document to a corresponding element in the IDOC format.

Example of Mapping a Business Document

The following diagram illustrates the process of mapping a business document for a two-action PIP.



Step	Description
1	Trading Partner A uses an outbound mapping service to map an internal business document 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 document is mapped, it is in a format that Trading Partner B's internal system can process.
4	Trading Partner B responds by using an outbound mapping service to map an internal business document to a RosettaNet PIP document.
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 internal system can process.

Creating an Outbound Mapping Service

In Software AG Designer, you can create an outbound mapping service using a flow service. This flow service must contain one or more MAP entities that map the business document generated by your internal system, 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 "Processing Large Business Documents" on page 123.

Assigning Input and Outputs

The input to the outbound mapping service is your internal business document, represented as an IData object.

Note: Typically, you invoke the wm.estd.common.util:getBizDocFromEvent service to retrieve the BizDocEnvelope, which contains your internal business document, and then invoke the wm.estd.common.util:recordFromBizDoc service to extract the business document content from the BizDocEnvelope. This process returns the business document as an IData object.

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

Note: If the documents for your internal system have DTDs, you can automatically import an external DTD in Designer to use as a starting point. Simply create a new external record and specify the source as `XML`, `DTD`, or `XML Schema`.

For instance, if you were mapping your internal system document to the `Pip3A4PurchaseOrderRequest` RosettaNet PIP document, the output of the mapping service would be `Pip3A4PurchaseOrderRequest(Pip3A4PurchaseOrderRequest)` in the *documents\Payload* IData object.

Important: The name and case of the IData object must match the IS document type exactly. If not, validation on the receiver's end will fail. In the above mentioned example, the receiver's RosettaNet-compliant server expects the name of the incoming RosettaNet PIP document to be `Pip3A4PurchaseOrderRequest`.

Flow Operations to Use

In the flow service, insert a MAP operation and use the service pipeline to map elements of the IS document type for your internal 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, wm.b2b.rn.rec.PIPs.

Example of an Outbound Mapping Service: Initiator (Sender)

If you are the initiator in a conversation and are implementing a PIP 3A4 Request Purchase Order, an outbound mapping service converts your internal business document to a RosettaNet PIP document (PO Request). For a working example, see the wm.b2b.rn.sample.company1.maps.outbound:PORequest service used in the PIP3A4 sample.

Example of an Outbound Mapping Service: Fulfiller (Receiver)

If you are the fulfiller in a conversation and are implementing a PIP 3A4 Request Purchase Order, the fulfiller returns a Purchase Order (PO) Acceptance to the initiator. Note that the `Pip3A4PurchaseOrderAcceptance` IData object that contains information from the fulfiller's internal system is mapped to the `Pip3A4PurchaseOrderAcceptance` IData object, or the RosettaNet PIP document in the *documents\Payload* IData object. For more information, see the wm.b2b.rn.sample.company2.maps.outbound:POAcceptance service used in the PIP3A4 sample.

Creating an Inbound Mapping Service

Use Designer to perform inbound mapping with flow services. An inbound flow service must contain one or more MAP operations that map fields from the incoming RosettaNet PIP document, through any intermediate steps, to the format of your internal business documents.

Note: For information about customizing your inbound mapping service to handle large documents, see "Processing Large Business Documents" on page 123.

Assigning Inputs and Outputs

The input to the inbound mapping service is a *userDocument* IData object.

userDocument IData Objects

The input IData object must be named *userDocument* . After validating the inbound RosettaNet PIP document, RosettaNet Module places this inbound document into the *userDocument* IData object. By doing so, RosettaNet Module makes the information immediately available without requiring an additional access call to data storage (thus improving performance).

Headers

You can retrieve headers from the pipeline using the pub.estd.rosettaNet:getRNOMimePart service, and then map the headers to your own IS document types that represent them. For more information about obtaining header objects and IS documents to which to map them, see pub.estd.rosettaNet:getRNOMimePart on page 96.

Note: If you migrated from RosettaNet Module 6.0.1, you can retrieve all the header IS document types from the pipeline. First, create a service that calls the pub.estd.rosettaNet:getRNOMimePart service multiple times for each required header object. Then, use that service at the start of the existing inbound map service.

Flow Operations to Use

In the flow service, 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 internal business document.

Example of an Inbound Mapping Service: Initiator (Sender)

If you are the initiator in a conversation and are implementing a PIP 3A4 Request Purchase Order, in the initiator's system, when RosettaNet Module 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* IData object, as an internal business document. For more information, see the wm.b2b.rn.sample.company1.maps.inbound:POAcceptance service used in the PIP3A4Sample.

Example of an Inbound Mapping Service: Fulfiller (Receiver)

If you are the fulfiller in a conversation and are implementing a PIP 3A4 Request Purchase Order, in the fulfiller's system, when RosettaNet Module 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, as an internal business document. For more information, see the wm.b2b.rn.sample.company2.maps.inbound:PORequest service used in the PIP3A4 sample.

Reusing Mapping Services

In 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 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.

If you are using different versions of the same PIP, you cannot reuse the same mapping service because the mapping service includes references to IS document types within a specific version of the PIP.

9 Customizing Process Model Template

Process Model Templates provided with webMethods RosettaNet Module

A *process model* is a diagram that represents a conversation (or business process). RosettaNet Module provides process model templates that you can use as a model to build your own process models.

In the process model, you can control the exchange of documents with your internal system by the services invoked at individual steps of the process model. When you customize the process model template, you can edit the services invoked by process model steps, replace the RosettaNet Dummy Doc Types with the TN XML document types that you have created for receive steps, specify inbound and outbound mapping services, and enhance the provided error and exception handling services, if necessary.

RosettaNet Module provides process model templates for different Integration Server versions to assist you in creating your own process models. Process model templates are located in the *Integration Server_directory*\packages\WmRosettaNet\ProcessModels directory.

Available Process Models

For users running RosettaNet Module on Integration Server version 7.x, the following process models are available:

- PipOneActionInitiatorModel.process
- PipOneActionResponderModel.process
- PipTwoActionInitiatorModel.process
- PipTwoActionResponderModel.process

For users running RosettaNet Module on Integration Server version 8.x, the following process models are available:

- PipOneActionInitiatorModel_8x.process
- PipOneActionResponderModel_8x.process
- PipTwoActionInitiatorModel_8x.process
- PipTwoActionResponderModel_8x.process

Use Designer to customize the process model templates based on the PIPs you are implementing. For more information about process models and Designer, see the Designer online help for your release. See “About this Guide” for specific document titles.

Required Process Model Modifications

To customize a process model template, you must modify the steps to correspond to your system processing. The following table lists the steps that you must modify, along with a description of the modifications you must make. All the below steps are applicable to both Initiator and Fulfiller.

Process Model Template Step	Description of Modification
Receive Internal Document	Subscribe to a TN XML document type.
Receive External Document	Subscribe to a TN XML document type.
Outbound Map	Add an outbound mapping service to map your internal document to a RosettaNet PIP document.
Inbound Map	Add an inbound mapping service to map your RosettaNet PIP document to an internal document.
Send to Back-End	Add a service to send the business document to the internal system.
Process ACK	Add a service to process the acknowledgement, for example, send the ACK to another department in your enterprise or trigger another service.
Notification of Failure Handler	Add a service to process the Notification Of Failure (NOF). For example, to respond with a receipt acknowledgment, add the service "pub.estd.rosettaNet:sendReceiptAck" on page 103. To respond with an exception document, invoke, "pub.estd.rosettaNet:sendException" on page 102.
Exception Handler	Add a service to handle the exception condition, for example, to send an e-mail message.

It is not mandatory to modify the **Error** step. But, if you modify the **Error** step, you must invoke the "pub.estd.rosettaNet:error" on page 96 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 must complete before customizing a process model template.

Prerequisite	For more information, see…
Define the trading partner profiles for the trading partners with whom you exchange business documents.	"Defining Trading Partner Profiles " on page 41 and Trading Networks administration guide for your release. See “About this Guide” for specific document titles.
Import the PIP Specification file or PIP archives you need, into Integration Server.	"Importing PIP Specification/PIP Archives" on page 49.
Create your inbound and outbound mapping services.	"Mapping a RosettaNet PIP Document" on page 71.
Create your *Send to Back-End* service.	"How Monitor Tracks Conversations and the ConversationID" on page 92 and "Customizing a Process Model Template" on page 82.
Create error and exception handlers, if necessary.	"Error and Exception Handling" on page 107.

Customizing a Process Model Template

To customize a process model template, you must first determine the appropriate process model template to use, based on the following criteria:

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

For example, PIP 3A4 Request Purchase Order requires either the initiator or fulfiller version of the two-action asynchronous process model template. To determine whether a PIP is the one-action or two-action model, visit the RosettaNet Web site (http://www.rosettanet.org), and look up the PIP specification that you imported.

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 Designer and import the required process model template from the directory: *Integration Server_directory* \packages\WmRosettaNet\ProcessModels. For help with this, or any step in this procedure, see Designer online help. See "About this Guide" for specific document titles.
2. For each step that waits for an action document, replace the RosettaNet Dummy Doc Type with the TN XML document type that you imported/created.

   For example, if you were using the PipTwoActionResponderModel.process template, you would replace the RosettaNet Dummy Doc Type in the **Receive Internal Document** and **Receive External Document** steps with the TN XML document types you imported/created for these steps.

   Important: You must select the property **Allow this step to start new process instance** for at least one receive step. For all the other steps, which must not start new process instances, unselect this property.

   Note: If you later change the name of a TN XML document type, you must also update the name of the TN XML document type in the process model, to avoid error.
3. Assign services as follows:
   - Assign outbound mapping and inbound mapping services to the **Outbound Map** and **Inbound Map** steps.
   - Assign error and exception handlers, if necessary.
   - Assign a service to the **Send to Back-End** step, that will send a business document to your internal system.

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

   Important: Some of the steps in the process model templates are set to invoke public services of RosettaNet Module. If you are customizing these steps to invoke your custom services, ensure that your custom service internally invokes the public service (originally invoked by the step).
4. Set timeout, retry count, and join-timeout values. For more information on timeout, retry count and join-timeout, see "Timeout, Retry Count and Join Timeout values" on page 84.
5. Modify step labels and icons as appropriate.
6. Save the process model.
7. Build and upload the process model.

When you build and upload a process model, a process model package (containing services and triggers) and a Process Engine script (or fragment) are created on Integration Server.

8. Using Monitor, enable the process model, if the process model is not already enabled. For instructions, see *webMethods Monitor User's Guide*.

Timeout, Retry Count and Join Timeout values

You can set timeout, retry count, and join timeout values in Designer.

- *Timeout* refers to the amount of time to wait, for any step in your process model to complete execution. For example, how long 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 executing a particular step. For example, how many times to attempt sending a Receipt ACK or 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. You can set the retry count value for each step independently of other steps. You must be careful when setting the retry count value for steps that send an action message to a trading partner. Setting the value of the **Step Retry Count** property in Designer changes the value for your PIP conversation, but, does not change the value in your trading partner's PIP conversation.
- *Join Timeout* refers to the timeout value for the steps that contain AND join or COMPLEX join. It can be a static value, a field value, or a business calendar value.

For more details about how to set timeout, retry count, and join timeout values, see the Designer online help for your release. See "About this Guide" for specific document titles.

Error Transitions

If you want a step in your process model to go to another step when an error occurs, use error transition.

For example, use error transition type to go from the *SendReceiptAck* step to *Wait for External Document* step, when acknowledgement delivery fails.

If you want *SendReceiptAck* step to transition to an error step after 10 retries, add a transition to *ExceptionHandler* step with *Transition Type* as *Step Iterations Exceeded*.

For more information on error transitions, see the Process Engine administration guide for your release. See "About this Guide" for specific document titles.

Defining Quality of Service (QoS) Settings in Process Model

Quality of service settings gives you a measure of reliability, visibility, and control. For more information on Quality of Service configurations you can set in Designer, see the Process Engine administration guide for your release. See “About this Guide” for specific document titles.

Express Pipeline QoS Setting

If you use the **Express Pipeline** QoS run time property in the process model, you must set the required input and output for all the steps in the model for RosettaNet transactions to complete successfully.

Optimize Locally QoS Setting

To use Synchronous PIPs that are supported only for RNIF 2.0 protocol, enable ‘Optimize Locally’ QOS run time property for the respective process model.

For information on the other QoS settings, see Process Engine administration guide for your release. See “About this Guide” for specific document titles.

Joining Back-end steps with response-wait steps

If a step is waiting for response from back-end, ensure that you add a transition between the executing step (for example, *Send to Back-End*) and waiting-step (for example, *Wait for InternalDocument*). If you do not add the transition, your processing will not be completed successfully. For a working example, see the PipTwoActionResponderModel process model template.

10 Running and Monitoring a Conversation

RosettaNet Conversation Flow

Conversations in RosettaNet Module consist of essentially three parts:

- Exchanging the business document between partners, such as a purchase order.
- System acknowledgments that indicate that messages were successfully received (or failed).
- Transforming business documents into the format used by the receiving system, for example from an internal format into a RosettaNet business object, or vice versa.

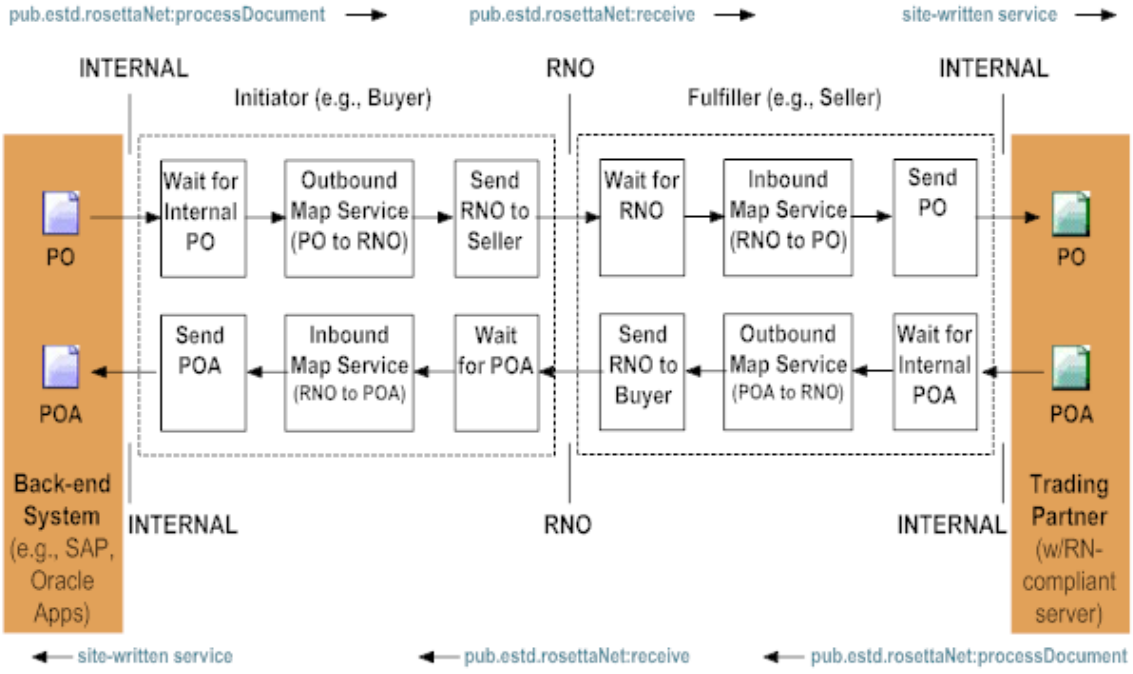
The process for sending and receiving documents is the same for both parties using RosettaNet Module, whether acting as the fulfiller or the initiator of the conversation:

- To send a business document generated by an internal system *to*RosettaNet Module, the pub.estd.rosettaNet:receive service is used.
- To send a business document *from*RosettaNet Module to the internal system, a custom-created service is used.

A unique *ConversationID* is assigned to each conversation that allows the system to track the exchange of documents and determine which acknowledgements belong to which business message.

Example of a RosettaNet conversation flow

This section provides an example of the conversation flow of a PIP3A4 document. The following diagram illustrates two distinct conversations: a conversation on the initiator's/buyer's side and a conversation on the fulfiller's/seller's side. The diagram 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 the pub.estd.rosettaNet:receive service.



Starting a Conversation

A conversation starts when the internal system on the initiator's side invokes pub.estd.rosettaNet:processDocument. An outbound mapping service must convert the business document received from the internal system into a RosettaNet object (RNO), that is the standard format used by both trading partners to exchange business documents. The outbound mapping service on the initiator's side includes a **Send to Seller** step. This step invokes a service that sends the document, now in RNO format, to the trading partner.

The conversation starts on the fulfiller's side when the pub.estd.rosettaNet:receive service, which has a **Wait for RNO** step, receives the document. The pub.estd.rosettaNet:receive service invokes wm.tn.doc:recognize service to recognize the document by matching it against the TN XML document types. Next, the fulfiller's custom-created mapping service converts the RNO document into the format used by the internal system. The **Send PO** step of the custom-service sends the business document to the internal system.

Both the pub.estd.rosettaNet:processDocument and pub.estd.rosettaNet:receive services invoke wm.tn.doc:recognize to recognize the document by matching it against the TN XML document types.

On each side, a BizDocEnvelope is created. Both the pub.estd.rosettaNet:processDocument and pub.estd.rosettaNet:receive services send the BizDocEnvelope to Process Engine.

Throughout this process, Process Engine determines if a *ConversationID* exists for the conversation. If not, Process Engine creates a new instance of the process model and assigns the *ConversationID* to the process model. Process Engine executes the conversation based on the steps defined in the process model.

Rejoining an Existing Conversation

Once the fulfiller's internal system has processed the purchase order, it creates the response. In this example, a Purchase Order Acceptance (POA) is created and the pub.estd.rosettaNet:processDocument service is called to convert the document into RNO format. The internal system always includes the original *ConversationID* in response documents so that the Process Engine can associate the document with the right conversation.

Process Engine searches for a running conversation (or process instance) where a **Wait for Internal POA** step is waiting for a POA and has a matching *ConversationID* . When Process Engine finds the correct conversation (or process), it rejoins the POA with the existing conversation. If Process Engine receives a POA, but cannot find the matching conversation, Process Engine ignores the POA.

On the initiator's side again, the pub.estd.rosettaNet:receive service recognizes the document, creates a BizDocEnvelope, and sends the BizDocEnvelope to Process Engine. Process Engine searches for a running conversation (or process instance) where a **Wait for POA** step is waiting for a POA with a matching *ConversationID* . When Process Engine finds the correct conversation (or process), it sends the POA to rejoin the existing conversation. The initiator's custom service then converts the document response into the internal system's format.

Sources of Conversation Status and Information

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

Source	Description
Monitor	My webMethods is the user interface for Monitor. From My webMethods, you can monitor and manage conversations (business processes). For more information about Monitor, see "About webMethods Monitor " on page 91, and *webMethods Monitor User's Guide*.
Trading Networks Transactions page	In My webMethods, query and analyze the documents exchanged through Trading Networks on the **Monitoring > Integration > B2B > Transactions** page. For information about transaction analysis, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Source	Description
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.

Why Monitor a RosettaNet Conversation?

You may want to determine the status of a conversation when your supplier or trading partner does not receive your purchase order. Monitor is a utility that allows you to monitor RosettaNet conversations to track the status of a transaction.

For example, let's say that you initiate a *PIP 3A4 Request Purchase Order* process for 30 PCs and send the PO to your trading partner. The next day, your trading partner states that the PO never arrived. In RosettaNet Module, you would be able to view the conversation history to determine the current status of the transaction and why it failed or was interrupted.

By monitoring the conversation, you can determine why your partner never received the PO and whether it was re-sent to the partner. By viewing the current status and progress of a RosettaNet transaction, you can take appropriate action, which might include retrying the conversation, editing your trading partner profiles, or editing your process models.

About webMethods Monitor

webMethods Monitor is a tool for managing process instances and conversations between trading partners. Using Monitor, you can do the following:

- 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.
- Retrieve information about a conversation and its execution (for example, status of the conversation or status and iteration of process model steps).
- View services used in the conversation.
- Perform control tasks to affect the state of a conversation, including suspending and resuming a conversation and editing and resubmitting process model steps.

Monitor accesses data from the Process Audit Log database component. For information about the Process Audit Log database, see " RosettaNet Module Run-Time Architecture and Components" on page 25.

The user interface for Monitor is My webMethods. For more information about how to perform these tasks, see *webMethods Monitor User's Guide*. For more information about archiving RNOs to the file system, see "Configuration Properties" on page 135.

How Monitor Tracks Conversations and the ConversationID

Every transaction and document that is exchanged through RosettaNet Module is known as a conversation and is assigned a unique *ConversationID* . Process Engine uses the *ConversationID* to monitor the progress of the "conversation" and the relationship of the PIP documents to each other. When a conversation starts, Process Engine chooses the appropriate process model to use and applies it to the conversation. Process Engine can run multiple instances of the same process model at one time. Each instance is a different conversation that Process Engine distinguishes by the unique *ConversationID* .

When Process Engine receives a document, it determines if the document belongs to an existing conversation by verifying whether the document has the same *ConversationID* as a running instance of a process model. A document enters your internal system, when the *Send to Back-End* step in the process model is triggered and Trading Networks submits the document to your internal system (the back-end). When this occurs, Process Engine sends information to the internal system, including the *ConversationID* .

For the conversation to continue properly, your internal system must keep track of the *ConversationID* and include it in the business document and all the responses that it returns. This means that, at design time, you must configure your internal system to maintain *ConversationIDs* when it receives business documents from the RosettaNet Module solution.

Note: The response document must always include the *ConversationID* so that webMethods RosettaNet Module can rejoin the document with the correct conversation.

ConversationID

The *ConversationID* is a concatenation of the initiator's Global Business Identifier (a DUNS number), a Process Instance Identifier, and the initiator focal role of the process model, separated by `-%%-`.

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 as *123456789-%%-1001-%%-Buyer* (where *Buyer* is the initiator focal role of the initial process model for PIP 3A4).

A Public Services in WmRosettaNet Package

Overview

This chapter describes the built-in services provided with the WmRosettaNet package to implement and support the RosettaNet-compliant functionality.

RosettaNet Module also uses services in the WmEstdCommonLib package. For information about these services, see *webMethods eStandards Modules Common Built-In Services Reference*.

Note: Public services are no longer available through WmRNIF20TRP and WmRNIF11TRP packages. See "Upgrading from RosettaNet Module 6.0.1" on page 33 for more information.

Summary of Elements in this Folder

The following elements are available in the public folder:

Element	Description
"pub.estd.rosettaNet:done" on page 96	Sets the current process (conversation) to a done (completed) state.
"pub.estd.rosettaNet:error" on page 96	Sets the current process (conversation) to an error state.
"pub.estd.rosettaNet:getRNOMimePart" on page 96	Retrieves a part of the RosettaNet document as an Integration Server document.
"pub.estd.rosettaNet:inboundValidation" on page 98	Invokes the appropriate inbound validation service.
"pub.estd.rosettaNet:migrate" on page 98	Migrates custom services that use the services in the WmIPRoot package version 6.x to the WmEstdCommonLib package version 7.1 and migrates references of WmIPRoot services

Element	Description
	to the services in the WmEstdCommonLib package.
"pub.estd.rosettaNet:processDocument" on page 99	Initiates the process model and starts a conversation.
"pub.estd.rosettaNet:processUnhandledDocument" on page 100	*Deprecated*. Sends a general exception to the trading partner who sent the unhandled document when a conversation receives an out-of-sequence document.
"pub.estd.rosettaNet:receive" on page 100	Receives inbound documents, recognizes them, and forwards them to the Process Engine.
"pub.estd.rosettaNet:send" on page 101	Invokes the appropriate service to send a document.
"pub.estd.rosettaNet:sendException" on page 102	Sends an exception message.
"pub.estd.rosettaNet:sendNOF" on page 102	Initiates a Pip0A1FailureNotification.
"pub.estd.rosettaNet:sendReceiptAck" on page 103	Sends receipt acknowledgment documents.
"pub.estd.rosettaNet:sendSynchronousResponse" on page 104	Sends a synchronous response. Supported for RNIF 2.0 protocol only.
"pub.estd.rosettaNet:verifyAcknowledgement" on page 105	Verifies incoming Acknowledgment documents.
"pub.estd.rosettaNet:waitStepInit" on page 106	Retrieves the BizDocEnvelope that a step waited for and

Element	Description
	TPA associated with the conversation.

pub.estd.rosettaNet:done

Sets the current process (conversation) to a done (completed) state.

Input Parameters

ProcessData	**Document Reference** An IData object that contains run-time process information for the process that needs to be set to a done state.

Output Parameters

None.

pub.estd.rosettaNet:error

Sets the current process (conversation) to an error state.

Input Parameters

ProcessData	**Document Reference** An IData object that contains run-time process information for the process that needs to be set to an error state.

Output Parameters

None.

pub.estd.rosettaNet:getRNOMimePart

A general purpose service that retrieves part of a RosettaNet document as an Integration Server document. This service can be used as follows:

- For incoming document processing. For example, as an exception handler. You should only use this service as an exception handler after a complete RosettaNet Object (RNO) is available. For example, after receiving an incoming RNO or after

sending an RNO when a complete RNO is prepared and data has been mapped to the format used by the internal system.

- To replace services when migrating from a previous RosettaNet release, such as wm.ip.rnif11.util:getRNIF11Documents and wm.ip.rnif20.util:getRNIF20Documents.

The output of this service is a document list to accommodate situations in which output consists of multiple documents, such as attachments.

Input Parameters

partName	**String** The part of the document to extract (for example, service header, preamble, or payload).
documentTypeName	**String** Optional. The fully qualified name of the IS document type, to maintain the order and dimensionality of elements of the resulting document.

Output Parameters

resultDocuments	**Document List** An array of objects that contain details of the part name provided in the input.

Usage Notes

You can map the output of this service to an IS document representing a particular part of an RNO, such as preamble. Obtain the IS document type using either of the following methods:

- If you are migrating from RosettaNet Module 6.0.1, you can use the document types available in 6.0.1. Import 6.0.1 documents into RosettaNet Module 7.1 SP2 using Designer and then map the imported documents.
- Create a new IS document type in Designer. Specify XML as the source and provide an XML document that contains header data, such as preamble header data from an RNO.
- Create the IS document type using the DTD provided by the RosettaNet standard for any specific header document type. webMethods RosettaNet Module provides DTDs specific to the transport protocols RNIF 1.1 and RNIF 2.0 in the following directories:
  - *Integration Server_directory*\packages\WmRNIF11TRP\dtd
  - *Integration Server_directory*\packages\WmRNIF20TRP\dtd

Map the output of this service to a document reference list corresponding to the IS document type you created earlier. For example, if you created a preamble IS document type, create a document reference list of preamble type and map this list to the *resultDocuments* output parameter of the service.

pub.estd.rosettaNet:inboundValidation

Validates the payload in the incoming RosettaNet document.

Input Parameters

bizEnv	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.
documents	**Document** An IData object that contains the documents associated with the conversation.
parameters	**Document** (optional) An IData object that contains the parameters associated with the conversation.

Output Parameters

sendFailed	**String** (optional) Returns true or false.
sendErrorMessage	**String** (optional) Returns the error message.

pub.estd.rosettaNet:migrate

Migrates custom services that use the services in the WmIPRoot package version 6.x to the WmEstdCommonLib package version 7.1 and migrates references of WmIPRoot services to the services in the WmEstdCommonLib package.

Input Parameters

listOfComponents	**String List** The list of components to be migrated.
backupLocation	**String** (optional) The backup location for the components you want to migrate. The default backup location is *Integration Server_directory*\RNBackup.

Output Parameters

errors	**String List** (optional) Returns the error messages.

pub.estd.rosettaNet:processDocument

Initiates the assigned process model and starts a conversation. This service executes the following steps:

- Recognizes the incoming document for its TN document type.
- Saves the incoming document to the Trading Networks database.
- Submits the recognized document (*bizdoc*) to the pub.prt.tn:handleBizDoc service.

You must enter *one* of the below parameters as input to this pub.estd.rosettaNet:processDocument service:

- *input*
- *node*
- *xmlString*

Input Parameters

input	**Document** The IData object from the pipeline, for example, *input/Pip3A4PurchaseOrderRequest* , that contains the incoming document to be processed as part of a new conversation or an existing conversation.
node	**Object** An XML node to process. The node must be an instance of com.wm.lang.xml.Node. You can add the XML node to the pipeline by using Integration Server Services. For example, the pub.xml:xmlStringToXMLNode service.
xmlString	**String** XML payload in the string format.
cid	**String** (optional) The conversation ID of the conversation that the document specified in *input* , *node* or *xmlString* is joining. If the document is not currently part of a conversation, leave *cid* null. **Note:** For more information on conversation ID and the format of conversation ID, see the "ConversationID" on page 92
filePath	**String** (optional) Path of the file for handling large document payload.
$reservation	**Object** (optional) A reservation object for handling a large document.

Output Parameters

cid	**String** (optional) The conversation ID of the conversation that the document specified in *input* , *node* or *xmlString* is joining, if one exists.

Usage Notes

This service can also be invoked via a TN Processing Rule, to initiate the process model.

pub.estd.rosettaNet:processUnhandledDocument

Deprecated- Sends a general exception to the trading partner who sent the unhandled document when a conversation receives an out-of-sequence document.

Input Parameters

bizdoc	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.

Output Parameters

None.

pub.estd.rosettaNet:receive

Receives inbound documents, recognizes them, and forwards them to the Process Engine.

Note: All trading partner profiles should use wm.ip.rn:receive instead of the default Trading Networks wm.tn.receive service.

Input Parameters

protocolInfo	**Document** An IData object that contains the protocol information on which the packet arrived.
contentStream	**Object** A java.io.InputStream to the received data.
contentInfo	**Document** An IData object that contains the received HTTP header information.

Output Parameters

exceptionCode	**String** (optional) Returns the exception code.
exceptionDescription	**String** (optional) Returns the error description.
exceptionComponentOrigin	**String** (optional) Returns the component name where the exception originated.

Usage Notes

As a security measure, the default ACL for this service is set to Administrators. You can change the ACL for this service using the RosettaNet Module parameter RosettaNet.receive.acl. For details, see "Configuring the Default ACL for a Receive Service" on page 138.

pub.estd.rosettaNet:send

Sends a document.

Input Parameters

bizEnv	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.
documents	**Document** An IData object that contains the documents associated with the conversation.
parameters	**Document** (optional) An IData object that contains the parameters associated with the conversation.
TPA	**Document Reference** An IData object that contains the TPA, trading partner profile information of the partners involved in the current conversation and process specific parameters.

Output Parameters

sendFailed	**String** (optional) Returns true or false.
sendErrorMessage	**String** (optional) Returns the error message.

isResponse	**String** (optional) Returns `true` if the request is synchronous. Returns `false` if the request is asynchronous.

pub.estd.rosettaNet:sendException

Sends an exception message.

Input Parameters

bizEnv	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.
documents	**Document** An IData object that contains the documents associated with the conversation.
parameters	**Document** (optional) An IDATA object that contains the parameters associated with conversation script.
TPA	**Document Reference** An IData object that contains the TPA, trading partner profile information of the partners involved in the conversation and process specific parameters.
reason	**String** (optional) The reason why an exception was generated.
exceptionType	**String** (optional) The type of exception (for example, general exception or ReceiptAcknowledgementException).

Output Parameters

sendFailed	**String** (optional) Returns true or false.
sendErrorMessage	**String** (optional) Returns the error message.

pub.estd.rosettaNet:sendNOF

Initiates a Pip0A1FailureNotification.

Input Parameters

bizEnv	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.
documents	**Document** An IData object that contains the documents associated with the conversation.
parameters	**Document** (optional) An IData object that contains the parameters associated with the conversation.
TPA	**Document Reference** An IData object that contains the TPA, trading partner profile information of the partners involved in the current conversation and process specific parameters.
FailureReason	**String** (optional) The reason for notification of failure.

Output Parameters

sendFailed	**String** (optional) Returns true or false.
sendErrorMessage	**String** (optional) Returns the error message.
isResponse	**String** (optional) Returns `true` if the request is synchronous and `false` if the request is asynchronous.

pub.estd.rosettaNet:sendReceiptAck

Sends receipt acknowledgment documents.

Input Parameters

bizEnv	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.
documents	**Document** An IData object that contains the documents associated with the conversation.
parameters	**Document** (optional) An IData object that contains the parameters associated with the conversation.

TPA	**Document Reference** An IData object that contains the TPA, trading partner profile information and process specific parameters for the current conversation.

Output Parameters

sendFailed	**String** (optional) Returns true or false.
sendErrorMessage	**String** (optional) Returns the error message.

pub.estd.rosettaNet:sendSynchronousResponse

Sends response synchronously. This service handles validation failures internally and sends an exception response synchronously. Supported for RNIF 2.0 protocol only.

You can also use this service to send a receipt acknowledgement synchronously by setting the *receiptAckResponse* input parameter to `true`.

Input Parameters

bizEnv	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.
node	**Object** (optional) An XML node to process. The node must be an instance of com.wm.lang.xml.Node. You can add the XML node to the pipeline by using Integration Server Services. For example, the pub.xml:xmlStringToXMLNode service.
xmlString	**String** (optional) XML payload in the string format.
filePath	**String** (optional) Path of the file.
$reservation	**Object** (optional) A reservation object for handling a large document.
documents	**Document** An IData object that contains the documents associated with the conversation.
parameters	**Document** (optional) An IData object that contains the parameters associated with the conversation.

TPA	**Document Reference** An IData object that contains the TPA, trading partner profile information and process specific parameters for the current conversation.
receiptAckResponse	**String** (optional) Whether to synchronously send a receipt acknowledgement (`true`) or a response document (`false`). The default is `false`.

Output Parameters

sendFailed	**String** (optional) Returns true or false.
sendErrorMessage	**String** (optional) Returns the error message.

pub.estd.rosettaNet:verifyAcknowledgement

Verifies incoming acknowledgment documents.

Input Parameters

bizEnv	**Document Reference** (optional) The BizDocEnvelope associated with the most recent document event.
documents	**Document** An IData object that contains the documents associated with the conversation.
parameters	**Document** (optional) An IData object that contains the parameters associated with the conversation.
TPA	**Document Reference** An IData object that contains the TPA, trading partner profile and process specific parameters for the current conversation.

Output Parameters

sendFailed	**String** (optional) Returns true or false.
sendErrorMessage	**String** (optional) Returns the error message.

pub.estd.rosettaNet:waitStepInit

Retrieves the BizDocEnvelope and the relevant TPA for a wait step. Any step in the process that waits for a document can invoke this service. This service does the following:

- Retrieves the BizDocEnvelope associated with the current document event
- Retrieves the TPA associated with the conversation, if it does not already exist

Input Parameters

None.

Output Parameters

documents	**Document** (optional) An IData object that contains the documents associated with the conversation.
roles	**Document** (optional) The roles information for the sender and receiver of the document.
parameters	**Document** (optional) An IData object that contains the parameters associated with the conversation.
bizEnv	**Document Reference** The BizDocEnvelope that is associated with the most recent document event.
TPA	**Document Reference** An IData object that contains the TPA, trading partner profile information and process specific parameters for the conversation.
sendErrorMessage	**String** (optional) Returns the error message.

B Error and Exception Handling

When Might Errors or Exceptions Occur?

Errors or exceptions might occur at any point in a RosettaNet PIP interchange, for example, 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, and so on).
- RosettaNet PIP documents are validated for structure and content.
- RosettaNet transactions expire after the retry limit is reached.
- RosettaNet PIP documents are received out of sequence, violating the document order specified in the particular PIP.
- Trading Networks's processing rule is involved in initiation of RosettaNet process, resulting in redundant persistence of documents.

General Errors With RosettaNet PIP Documents

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

RNIF Version	Error	How RosettaNet Module Responds to the Error
Inbound documents		
1.1, 2.0	Header validation fails	Process Engine puts the conversation in an error state. Sends an exception with the error reason to the trading partner.
1.1, 2.0	Signature verification fails	Process Engine puts the conversation in an error state. Does not send an exception.
1.1, 2.0	Content validation fails	Process Engine puts the conversation in an error state. Sends an exception with the error reason to the trading partner.

RNIF Version	Error	How RosettaNet Module Responds to the Error
2.0	Decryption fails	Process Engine puts the conversation in an error state. Does not send an exception.
2.0	Unpacking a RNO when transport protocol is in Test mode	Process Engine puts the conversation in an error state. Sends an exception with the error reason to the trading partner.
Outbound documents		
1.1, 2.0	Validation fails	Process Engine puts the conversation in an error state. Does not send an exception. (This error can occur only if outbound validation is turned on.)
1.1, 2.0	Cannot package the RNO	Process Engine puts the conversation in an error state. Does not send an exception.
1.1, 2.0	Error occurs during HTTP/HTTPS transmission	RosettaNet Module retries the transmission the designated number of times. If the transmission fails on all attempts, Process Engine puts the conversation in an error state.

Transaction Time Out Errors: Notification of Failure (NOF)

When a transaction with your trading partner expires or times out, a PIP 0A1 Notification of Failure can be initiated. To initiate NOF, add the invocation of NOF on timeout in your process model. PIP 0A1 is designed to communicate between trading partners when errors occur at the various stages of processing. This informs both trading partners that a problem occurred and where to look for the issue. To illustrate how a Notification of Failure works, the following example explains what happens when there is a failed conversation.

Acting as the initiator, you send your initial request to your partner and receive a Receipt Acknowledgement (ACK). The conversation waits for a Response document from your trading partner, but none arrives. If no response is received within the specified timeout time, the conversation in your enterprise expires.

When a conversation expires in your enterprise, your system will initiate the PIP 0A1, as defined in your process model, to notify your trading partner that you never received a response and there may be a problem in the trading partner's system.

RosettaNet Module sends a PIP 0A1 Notification of Failure using the alternate delivery settings defined in the Trading Partner profile. This alternate delivery method is used

instead of the preferred delivery method whenever there is a communication failure, for example, when your partner's Web server is down. If no alternate delivery method is defined in the trading partner's profile, RosettaNet Module uses the preferred delivery method. If you did not specify a preferred delivery method, an error occurs.

Note: In webMethods RosettaNet Module 6.0.1, NOF uses v1.0 and v02.00 versions of PIP 0A1, where v1.0 is used for RNIF 1.1 and v02.00 is used for RNIF 2.0. Now, RosettaNet standard organization provides only v02.00 version of PIP 0A1. As a result, in webMethods RosettaNet Module 7.1 SP2, NOF uses only v02.00 of PIP 0A1 for both RNIF 1.1 and RNIF 2.0.

Sending NOF Documents

To send an NOF to your trading partner, you must define a way to trigger the NOF. In other words, you must define your PIP process models to initiate a PIP 0A1 when error conditions are encountered during processing. In addition, your trading partners must also configure their RosettaNet Module installations to accept NOFs.

To send the Notification of Failure (NOF) document

In Integration Server Administrator, go to RosettaNet page and import the SamplePIP0A1.par file, available on the Communities Website: http://communities.softwareag.com/ecosystem/communities/public/Developer/webmethods/codesamples/.

Note: Import the sample specific to the version of IS you are using.

This imports all NOF related assets, including Trading Networks XML document types, IS documents and TPAs. For more details about importing PIP archives, see "Importing PIP Specification/PIP Archives" on page 49.

Note: For working example of the NOF usage scenario, check the SamplePIP3A4 Buyer model, part of the RosettaNet samples available on the Communities Website.

Redundant Document Persistence Errors

When Trading Networks's processing rule is used to initiate the RosettaNet process, some of the internal documents gets persisted multiple times in the Trading Networks database. This can happen because:

- Both the Process Engine and Trading Networks saves the same document in Trading Networks database. Trading Networks saves the document, say DocA without a *ConversationID* and Process Engine saves the same document DocA with *ConversationID* added to the document.

- Same outbound message gets sent multiple times when the Trading Networks's processing rule is involved in the initiation of RosettaNet process.
- pub.client:http service is used for posting the internal document to wm.tn:receive service.

Resolving redundant persistence of documents by Process Engine and Trading Networks

A document may be saved twice, once with a *ConversationID* and second time, without a *ConversationID* . You can delete the document which does not contain the *ConversationID* . To resolve the redundant persistence of the internal document (that does not contain ConvesationID) in the Trading Networks database, change the existing property of the TN XML Document Type which is getting persisted redundantly.

To resolve redundant persistence of internal documents

1. In My webMethods, open the TN Document Type which is being persisted redundantly.
2. Go to Options tab.
3. Remove selection for saving the document to database.

Resolving duplicate sending of outbound messages

To prevent sending the same outbound message multiple times:

Add a string variable named *prtIgnoreDocument* with value set as `true` in the pipeline. This variable must be added before invoking the TN services that invoke the processing rule initiating the RosettaNet process.

Resolving redundant document persistence when using pub.client:http service

To avoid redundant document persistence when pub.client:http service is used for posting the internal document to wm.tn:receive service, do the following:

In My webMethods, go to the Processing Rule defined for the internal document. Execute the processing rule asynchronously.

Handling Errors and Exceptions

RosettaNet Module provides handler services for performing 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 these handler services or create your own.

What is a Handler Service?

A handler service is a service that a process model step invokes when triggered by an event in the PIP process. For example, in the process model template, the *Verify ACK* step invokes the "pub.estd.rosettaNet:verifyAcknowledgement" on page 105 service when the initiator receives a receipt acknowledgement (ACK) from the fulfiller.

In Designer, 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 RosettaNet Module

The following table describes the handler services that RosettaNet Module provides. For more information about any of these services, see the "Public Services in WmRosettaNet Package" on page 93.

Service	Description
"pub.estd.rosettaNet:done" on page 96	Sets the last step in a conversation to completed.
"pub.estd.rosettaNet:error" on page 96	Puts a RosettaNet conversation in an error state.
"pub.estd.rosettaNet:inboundValidation" on page 98	Validates inbound RosettaNet business documents.
"pub.estd.rosettaNet:processUnhandledDocument" on page 100	Handles a RosettaNet PIP document that does not conform to any TN XML document type.
"pub.estd.rosettaNet:send" on page 101	Sends RosettaNet PIP documents to a trading partner using the appropriate transport protocol.

Service	Description
"pub.estd.rosettaNet:sendReceiptAck" on page 103	Sends the receipt ACK upon receipt and validation of a RosettaNet PIP document.
"pub.estd.rosettaNet:sendSynchronousResponse" on page 104	Sends receipt acknowledgement or response document synchronously. Supported for RNIF2.0 protocol only.
"pub.estd.rosettaNet:sendNOF" on page 102	Initiates a PIP0A1 Notification of Failure and invokes the appropriate startNOF service based on the transport protocol used by the conversation.
"pub.estd.rosettaNet:verifyAcknowledgement" on page 105	Invokes the appropriate verifyAcknowledgement service based on the transport protocol used by the conversation.

Note: The above handler services are not recommended for editing.

Why initiate an Error Process manually?

You may want to manually initiate an error process to:

- **Void an already completed transaction.** For example, suppose that, as the initiator, you 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 from you, indicating that you have received their Purchase Order Acceptance (POA). However, after the conversation ends, you encounter a semantic error in that partner's POA (for example, the POA states that you ordered four PCs instead of only two, as you stated in your PO Request).
- **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 internal system, you realize that you have a semantic problem with the PO received.

Saving Failed Documents to Trading Networks Database

If an error or exception occurs during the RosettaNet transaction, the document may be lost, without being saved in the database. To prevent this, two TN XML Document Types - RN11BizDocType (for RNIF Version 1.1) and RN20BizDocType (for RNIF Version 2.0) are added as placeholders in the Trading Networks database.

When an error occurs, the information present in the message under processing, is saved to the placeholder document. The **Failure Reason** is also added to the placeholder document.

You can access the saved information present in the Trading Networks database through My webMethods. The failed documents are saved in the Trading Networks database along with the time at which the error occurred. If multiple errors occur during the RosettaNet transaction, you can distinguish between the documents using the timestamp associated with the document.

By default, the Processing Status of the failed document is set as NEW in the Trading Networks database.

Setting processing status as ERROR

The Processing Status of the saved failed documents is set as NEW by default. You can configure to set the processing status as ERROR.

To set processing status as ERROR

1. Go to the directory *Integration Server_directory* \packages\WmRosettaNet\config and open the config.cnf file.
2. Set the value of *RosettaNet.FailedDocument.ChangeProcessingStatus* as `true`.

   If *RosettaNet.FailedDocument.ChangeProcessingStatus* is not present in the config.cnf file, add *RosettaNet.FailedDocument.ChangeProcessingStatus* =`true`.
3. Reload the WmRosettaNet package.

The Processing Status of subsequent failed documents will be set as ERROR.

Where to Find Information About Errors

You can view information about errors with transactions and conversations on the My webMethods pages that contain Trading Networks and Monitor data. For more information about these resources, see "Sources of Conversation Status and Information" on page 90.

RosettaNet Module Error Codes

0002.000000.000003E TPA cannot be null. Specify valid TPA.

Explanation: TPA is not defined for the transaction.

Action: Define and enable the TPA using My webMethods Server.

0002.000000.000006E Valid security certificate not set in Trading Partner's profile.

Explanation: The trading partner profile in Trading Networks is not configured with valid certificates.

Action: Configure a valid certificate in Trading Networks partner profile.

0003.000000.000001E Cannot invoke service '{0}' in service group '{1}'.

Explanation: There was an issue executing the specified service.

Action: Contact Software AG Global Support.

0003.000000.000002E Cannot create the XML document.

Explanation: There was an error while creating the header of the RosettaNet message.

Action: Contact Software AG Global Support.

0003.000000.000003E Cannot find hostname required to create Content ID.

Explanation: Unable to retrieve the host name of the machine. As a result, the contentID field does not contain host name. This will not affect the functioning of the module.

Action: No action required.

0003.000000.000004E RosettaNetException occurred while checking the transport type.

Explanation: Could not verify the type of RNIF transport in the received RosettaNet message.

Action: Check whether the WmRNIF20TRP package is installed and enabled. Also, verify that the HTTP header of the received RosettaNet message contains correct transport related information.

0003.000000.000006E RosettaNet message sending failed.

Explanation: There was an error while sending the rosettanet message. See the server log for details.

Action: Refer to the server log for details.

0003.000000.000007E Error occurred while validating the RosettaNet message. See server log for details.

Explanation: Error occurred when validating the received RosettaNet message. The validation error is logged in the server log.

Action: Refer to the validation error in the server log.

0003.000000.000011E Cannot find the RosettaNet header: '{0}'.

Explanation: The received RosettaNet message does not have the mentioned header.

Action: The received RosettaNet message is invalid. Correct the message according to Rosettanet specification.

0003.000000.000012E RosettaNetException occurred while creating the exception message.

Explanation: There was an error while creating or sending the RosettaNet exception message.

Action: Refer to the server log for details.

0003.000000.000013E Found an invalid header at '{0}:{1}'. The invalid header value in the incoming stream is: '{3}'. Valid header value is: {2}.

Explanation: The received RosettaNet message has an invalid header.

Action: Refer to the server log for details.

0003.000000.000014E Found a header at '{0}:{1}' with invalid length. The invalid header value in the incoming stream is: '{3}'. Valid length is between: {2}.

Explanation: The header in the received RosettaNet message has invalid length.

Action: Refer to the server log for more details.

0003.000000.000015E The ReceiptAcknowledgment message requires non-repudiation. Ensure that the message is signed.

Explanation: Signing is enabled for sending the acknowledgement in the TPA, but the received message is not signed.

Action: Ensure whether non repudiation is required between partners and update the TPA accordingly.

0001.000000.000002E Cannot find the MIME header '{0}' in '{1}'.

Explanation: The received RosettaNet message does not contain the required MIME header.

Action: Refer to the server log for further details. Check with the partner with the error details in the server log.

0001.000000.000003E TPA cannot be null. Specify valid TPA.

Explanation: TPA is not defined for the trading partner.

Action: Define and enable the TPA using My webMethods.

0001.000000.000005E Set the encryption algorithm in the TPA, from Trading Networks Console / My webMethods.

Explanation: Valid encryption algorithm is not set in the TPA.

Action: Select the appropriate encryption algorithm using My webMethods for the TPA in RNIF2.0/EncryptionAlgorithm.

0001.000000.000006E Missing security information {0}.Go to Trading Networks Console / My webMethods and set a valid security certificate in the trading partner's profile.

Explanation: The trading partner's profile does not have a valid certificate configured.

Action: Use My webMethods Server to configure a valid certificate in trading partner's profile.

0001.000000.000007E Error occurred while parsing the RNO MIME message.

Explanation: There was an error while parsing the received RosettaNet message.

Action: Refer to the server log for more details.

0001.000000.000011E The input document '{0}' must be an XML document.

Explanation: The specified TN business document is invalid.

Action: This is an internal error. Contact SoftwareAG Global support.

0001.000000.000012E Unknown input document. Define the document type using Trading Networks Console / My webMethods.

Explanation: The specified document type is not defined in Trading Networks.

Action: Ensure that the document type is created and is enabled in trading networks, using Trading Networks Console or My webMethods.

0001.000000.000013E Unknown input document '{0}'. Define the document type using Trading Networks Console / My webMethods.

Explanation: The specified document type is not defined in Trading Networks.

Action: Ensure that the specified document type is defined and enabled in Trading Networks, using TN Console or My webMethods.

0001.000000.000014E Cannot recognize the incoming payload with any xml doc type.

Explanation: The doctype for the RosettaNet message is not present in Trading Networks.

Action: Either create the specified doctype in My webMethods, or correct the RosettaNet message to specify the appropriate root tag.

0001.000000.000016E Cannot load the TPA that has sender={0}, receiver={1}, and TPA ID = {2}.

Explanation: There is no TPA configured for the specified Sender, Receiver and Agreement ID.

Action: Create a TPA with the specified Sender, Receiver and AgreementID.

0001.000000.000019E Trading Networks cannot encrypt the document. Check the certificate settings in the trading partner profile.

Explanation: The profile in Trading Networks has not been configured with valid certificates.

Action: Configure a valid certificate in the trading partner's profile.

0001.000000.000020E Cannot find the certificate '{0}'. Set the security certificate key in the trading partner profile using Trading Networks Console / My webMethods.

Explanation: Could not find the private key in the configured certificate. An invalid certificate has been configured in Trading Networks.

Action: Specify a valid certificate in Trading Networks.

0001.000000.000021E Cannot detect the RNIF transport protocol. Check if required RNIF headers are present in the incoming message, and the required RNIF packages are loaded and enabled.

Explanation: Unable to detect the RNIF transport protocol. The required RNIF headers may be missing in the incoming message, or the required RNIF packages are not loaded and enabled.

Action: Check if the required transport package (WmRNIF11TRP or WmRNIF20TRP) is installed and is enabled and also verify that, the received HTTP Header of the RosettaNet message has the correct transport related information.

0001.000000.000031E No specified delivery method for the trading partner. Define the delivery method in the trading partner's profile using Trading Networks Console / My webMethods.

Explanation: There is no delivery method specified for the trading partner.

Action: Go to Trading Networks Console / My webMethods and define the delivery method in the trading partner's profile.

0001.000000.000032E Preferred protocol not set for the trading partner. Specify the preferred protocol in the trading partner's profile using Trading Networks Console / My webMethods.

Explanation: There is no preferred protocol for the trading partner.

Action: Specify the preferred protocol in the trading partners profile from Trading Networks Console / My webMethods.

0001.000000.000034E Data integrity failure: Cannot find '{0}'.

Explanation: The specified variable is not present in the pipeline.

Action: Check if the specified variable is dropped from the pipeline.

0001.000000.000035E Cannot find the process model with model ID {1} for the TN document '{0}'. Ensure that the model is saved and active.

Explanation: Cannot find the process model for the specified TN document.

Action: Ensure that the model for the TN document is saved and enabled.

0001.000000.000036E Failed to load the input TPA. This may happen if the TPA is not in a 'Agreed' status or the conversation id to which this TPA belongs is not valid, the conversation id for this PIP is either expired or is not valid.

Explanation: Failed to load the specified TPA. This may happen if the TPA is not in a 'Agreed' status or the conversation id to which this TPA belongs is not valid, the conversation id for this PIP is either expired or is not valid.

Action: Ensure that the Status of TPA is set as 'Agreed'.

0001.000000.000039E Cannot find TPA that contains the sender {0}, receiver {1}, and agreement ID {1}. Define the TPA using Trading Networks Console / My webMethods.

Explanation: Cannot find a trading partner agreement (TPA) for the specified sender, receiver and agreement ID.

Action: Using TN Console / My webMethods, define the trading partner agreement.

0001.000000.000040E The required attribute '{0}' is not defined for extraction in the TN document type '{1}'. Define the attribute '{0}' to extract using Trading Networks Console / My webMethods.

Explanation: The specified attribute is not defined in the TN document type.

Action: Use Trading Networks Console / My webMethods to define the specified attribute.

0001.000000.000041E TPA '{0}' does not contain the required field 'IS Schema'. This error occurs when the schemaValidation parameter is not set to true and the 'IS Schema' value is not set in the TPA. Set the 'IS Schema' parameter for the TPA using Trading Networks Console / My webMethods Server.

Explanation: This error occurs when the schemaValidation parameter is not set to true and the 'IS Schema' value is not set in the TPA.

Action: Use Trading Networks Console / My webMethods to set the 'IS Schema' parameter for the specified TPA.

0001.000000.000046E Payload validation failed. Cannot find PIP '{0}' in package WmRNPIPs. Ensure that the required PIP schema or DTD has been imported.

Explanation: Payload validation failed because RosettaNet module could not find the required ISDoc in WmRNPips package.

Action: Ensure that the required PIP has been imported using PIP tools or using the par import feature.

0001.000000.000048E Payload validation failed for document type '{0}' with schema {1}. The following errors occurred: {2}.

Explanation: Payload validation failed for the specified ISDocument or IS schema.

Action: Check the server log for the validation failure reason.

0001.000000.000049E The incoming message contains an invalid signature.

Explanation: The signature verification failed for the received message. The Rosettanet message has an invalid signature.

Action: Check the validity of the partner certificates, or check with the partner and request to resend with valid signature.

0001.000000.000056E Error occurred while decrypting the message.

Explanation: There was an issue while decrypting the received Rosettanet message.

Action: Refer the server log for further details about the error.

0001.000000.000057E Partner profile of '{0}' that has the partner ID '{1}' and partner name {2} is inactive. Activate the partner using Trading Networks Console / My webMethods.

Explanation: The specified partner profile is inactive.

Action: Use TN Console / My webMethods to activate the partner profile.

0001.000000.000058E TPA validation for the transport type failed. Received an invalid RNO {1} instead of RNO '{0}'. Update the transport type using Trading Networks Console / My webMethods.

Explanation: The version of the received Rosettanet message is not in sync with version specified in the TPA.

Action: Ensure that the Rosettanet message for the given RNIF transport version is same as what is defined in the TPA.

0001.000000.000059E The received request does not contain signature, but the TPA for this request requires that the request should contain signature. If you do not require that this partner sign the request, go to Trading Networks Console / My webMethods and update the required TPA parameter.

Explanation: The received Rosettanet message does not contain signature. The TPA for this request requires that the request should contain signature.

Action: If you do not need the partner to sign the request, go to TN Console / My webMethods, and update the required TPA parameter for signing.

0001.000000.000079E Validation service failed with following error '{0}'.

Explanation: There was an error while executing the custom validation service specified in the TPA.

Action: Refer to the server log for details about the error.

0001.000000.000080E Found an invalid header at '{0}:{1}'. The invalid header value in the incoming stream is: '{3}'. Valid header value is: {2}.

Explanation: The received Rosettanet message has an invalid header.

Action: Refer to the server log for the error details.

0001.000000.000081E Trading Networks cannot decrypt the document. Check for the certificate settings in the trading partner profile.

Explanation: There was an error while decrypting the received Rosettanet message.The profile certificate is either not specified or is invalid.

Action: Use trading networks to specify a valid certificate for the profile.

0001.000000.000083E Found a header at '{0}:{1}' with invalid length. The invalid header value in the incoming stream is: '{3}'. Valid length is between: {2}.

Explanation: The length of the header of the received Rosettanet message is invalid.

Action: Refer to the server log for details about the error.

0001.000000.000084E Required payload encryption does not exist in the incoming document.

Explanation: The received Rosettanet payload is not encrypted, but the TPA is configured for payload encryption.

Action: Correct the TPA data to disable payload encryption or request the partner to send encrypted payload.

0001.000000.000085E Required header encryption does not exist in the incoming document.

Explanation: The header of the received Rosettanet message is not encrypted, but the TPA is configured for header encryption.

Action: Correct the TPA data to disable header encryption or request the partner to send encrypted header.

0001.000000.000086E Insufficient MIME components. Expecting at least 3 mime components, but received only '{0}'.

Explanation: The received Rosettanet message should have atleast 3 mime components, but this message has less than 3.

Action: Request partner to send a valid Rosettanet message with required MIME components.

0001.000000.000087E MIME components are not in proper sequence. Ensure that sequence is as follows: Preamble Header, Service Header and Service Content.

Explanation: The headers of the received Rosettanet message are not in the required order as per specification.

Action: Request the partner to send valid Rosettanet message with correct order of MIME components.

0001.000000.000088E MIME components are not in proper sequence. Ensure that sequence is as follows: Preamble Header, Delivery Header, Service Header and Service Content.

Explanation: The MIME components of the received Rosettanet message are not in correct order as per specification.

Action: Request the partner to a send a valid Rosettanet message.

0001.000000.000091E Transport Registry found null. Exception reason being - '{0}'

Explanation: Could not find the transport package while trying to send an exception message to the partner.

Action: Ensure that either one of the transport packages RNIF11TRP or RNIF20TRP are present and are enabled.

0001.000000.000092E RNIF 1.1 does not support Attachment with Payload. Remove attachment and retry.

Explanation: RNIF 1.1 does not support attachments, but the Rosettanet message has attachments.

Action: Correct the message and retry.

0001.000000.000093E Unable to make transport request. The response code was: {0} and status message was: {1}

Explanation: There was a HTTP error while sending Rosettanet Message.

Action: Refer to the server log for more details about the error code and message.

0001.000000.000094E Unable to make transport request. Please check the host details.

Explanation: There was a HTTP error while sending Rosettanet message.

Action: Refer to the server log for more details about the error code and message.

0001.000000.000096E Retries of large document payloads is not supported. Re-run the process.

Explanation: Retry of sending Rosettanet message having large document payloads is not supported.

Action: Resend the Rosettanet message.

0001.000000.000098E Active process model not found for business document '{0}'. Ensure that the required process model is built and uploaded.

Explanation: There is no active process model for the specified Rosettanet document.

Action: Ensure that the required process model is built, uploaded and is active.

C Processing Large Business Documents

Overview

The default document processing features in RosettaNet Module handles RosettaNet PIP documents of size up to 15 MB, on an average (1 GB to 1.5 GB memory) installation. If your PIP documents are greater than 15 MB in size, you must do some configuration settings in Integration Server and Trading Networks.

Most large XML documents contain some non-repeated information and a large amount of repeated sections of XML. For example, a PIP 3D8 document contains repeated sections of the *workInProcess* line item.

Integration Server provides a set of services, such as the pub.xml:getXMLNodeIterator service, to aid in processing large, repetitive XML documents. These services allow you to convert the repeating sections of large XML documents into IData objects. This enables Integration Server to process each *workInProcess* line item of a PIP 3D8 document, individually, rather than all at once. For more information about the pub.xml:getXMLNodeIteratorservice, see the Integration Server built-in services reference guide for your release. See “About this Guide” for specific document titles.

Configurations to Process Large Business Documents

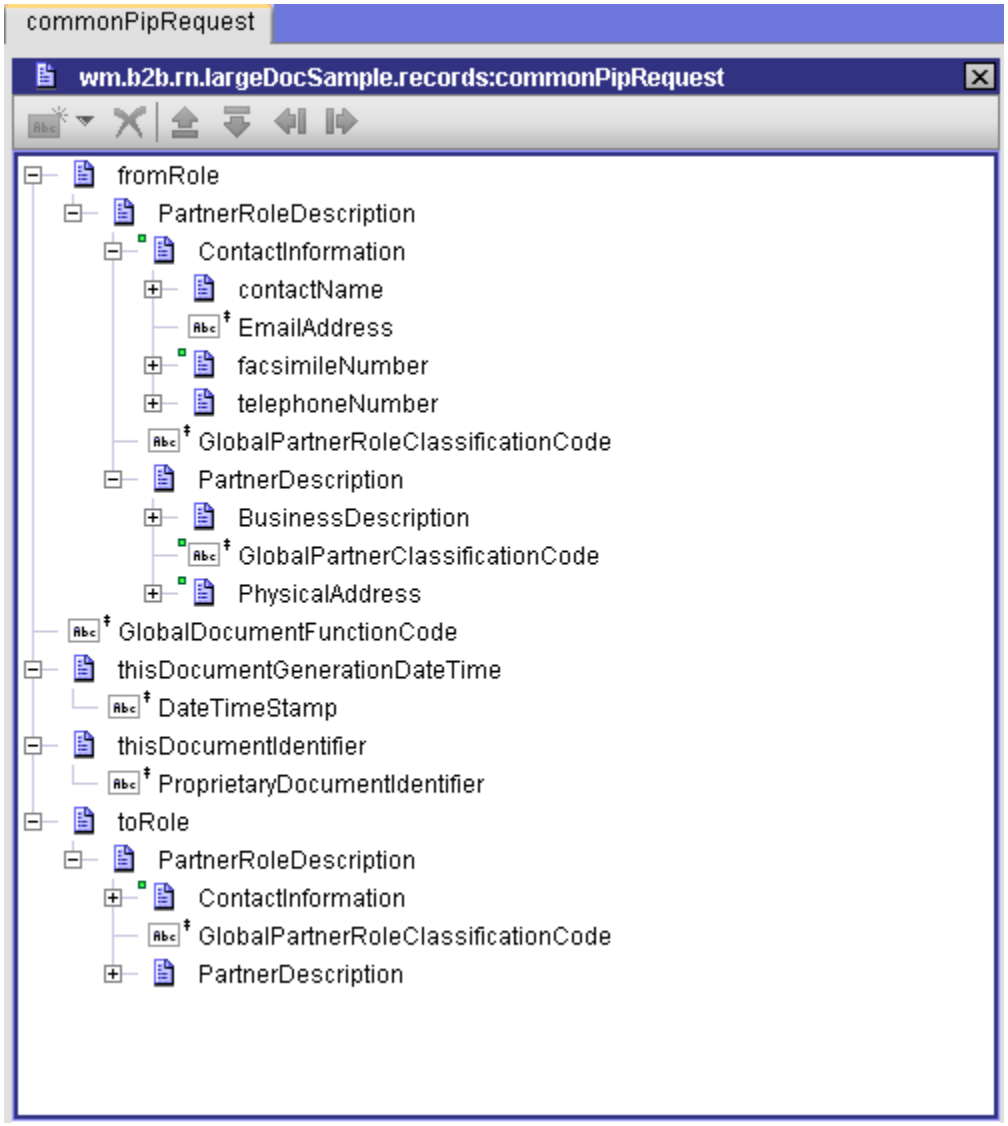
To enable RosettaNet Module to process large XML documents, during design time you must customize your PIP, Mapping Services, and Validation Services.

The remainder of this chapter explains how to customize PIPs and the mapping and validation services to enable RosettaNet Module to process large business documents, using the PIP 3D8 Large Document Sample as an example. For more information about mapping services, see "Mapping a RosettaNet PIP Document" on page 71.

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

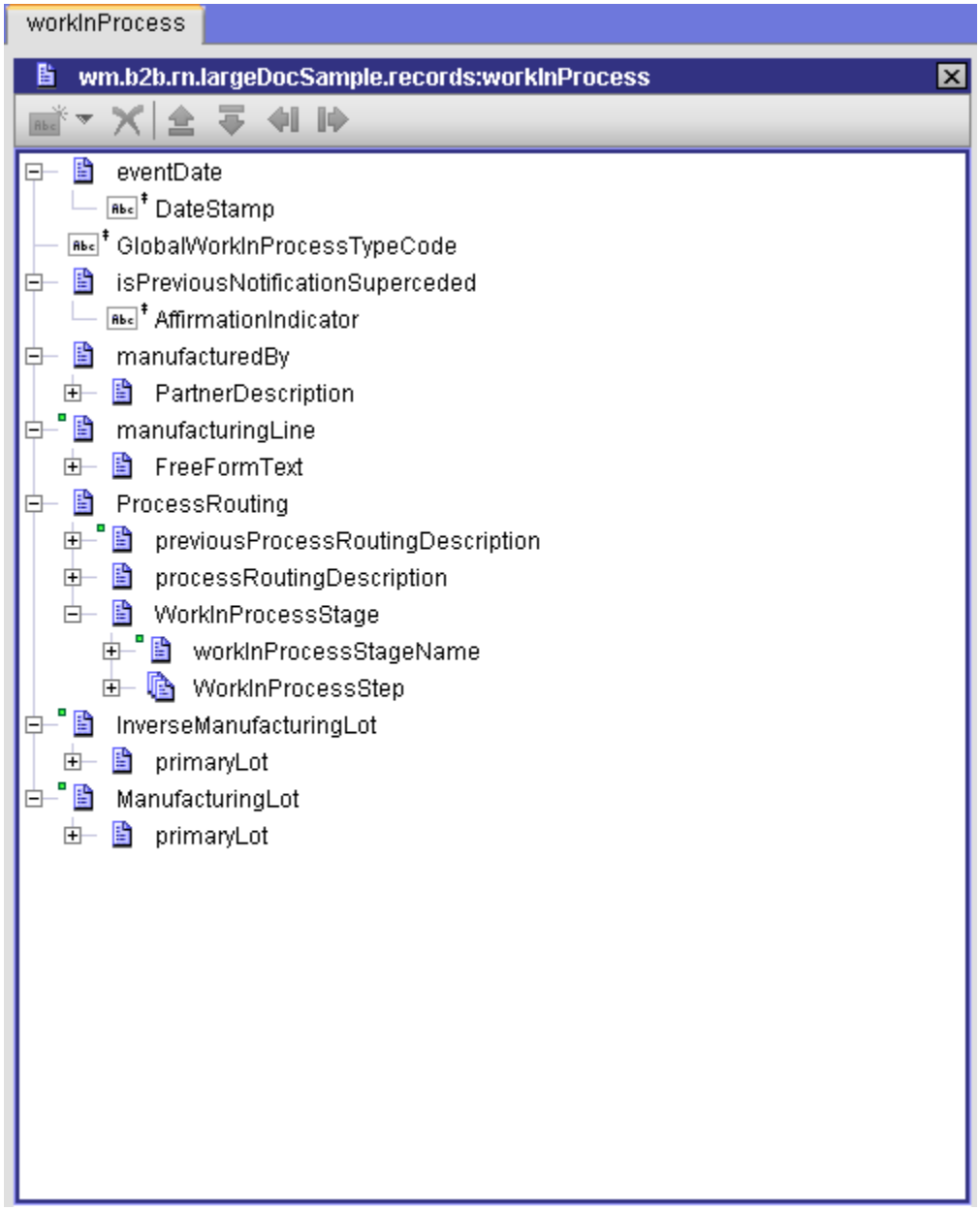
To process a large RosettaNet PIP document, first, you must separate the elements into non-repeating and repeating parts. This separation is required so that RosettaNet Module can validate the PIP in multiple chunks.

In the following figure, the non-repeating part, commonPipRequest, has been split from PIP 3D8.



The commonPipRequest section contains information that would only appear once in a PIP 3D8 document (for example, sender and receiver contact information, descriptions, and Global Partner Classification Codes).

The following figure illustrates the repeating parts of PIP 3D8, workInProcess.



The workInProcess section contains information that repeats in a PIP 3D8 for multiple product components (for example, process routing, manufacturing line, and work-in-process stage of a particular component).

Step 2: Configure the Temporary Disk Space (TSpace)

Trading Networks temporarily stores large documents in hard disk drive space rather than in memory. This temporary disk space is referred to as TSpace. You can configure the location of this temporary disk space, the maximum number of bytes that can be stored at any one time, and the minimum number of milliseconds that the InputStream should be allowed to write to TSpace before the corresponding file is deleted.

To configure TSpace in Integration Server

1. In Integration Server, select **Settings** > **Extended Properties**, edit **Extended Settings.**
2. Define the following parameters:
   - *watt.server.tspace.location* —The directory where Trading Networks should store document.
   - *watt.server.tspace.max* —The maximum number of bytes that can be stored at any one time in the hard disk drive space.

     Note: The required size of the hard disk drive space, to temporarily save documents, depends on the following:
     - Number of documents processed concurrently.
     - Size of documents, for example, size of the RosettaNet payload processed.

     Set *watt.server.tspace.max* as a value that is at least 15 times greater than the combined size of documents being processed. For example, if the size of a typical payload is 50 MB, and 10 such payloads are concurrently processed, set the value of *watt.server.tspace.max* as 50*10*15 MB, in bytes. Also, ensure that the specified amount of space is available on your hard disk.
   - watt.server.tspace.timeToLive—The number of milliseconds that Integration Server should temporarily store documents in TSpace rather than in memory. If you set this parameter to 180000 milliseconds, documents are deleted three minutes after they are created. A value of 0 indicates that documents are deleted the next time a reservation file is created. Setting this parameter prevents the instantaneous deletion of files and exceptions that occur while reading back from the file.

   For more information about setting these parameters, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.
3. Save the changes.
4. Restart Integration Server.

Step 3: Configure Trading Networks properties

You must configure the below explained Trading Networks properties to enable RosettaNet module to process large business documents.

To configure Trading Networks properties related to large document processing in RosettaNet module

1. Open the Trading Networks properties.cnf file and set the required values for the following properties:

   - *tn.BigDocThreshold*
   - *tn.xml.xqlThreshold*

   For information on configuring the Trading Networks properties, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

2. Reload the WmTN package.

   For information on reloading the WmTN package, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

Step 4: Customize the Outbound Mapping Service

You can either customize an existing mapping service or create a new one to handle large business documents. The following procedure describes how to customize an existing outbound mapping service.

To customize the outbound mapping service

1. In Designer, invoke the wm.estd.common.tspace:createFile service to open a reservation in TSpace. This service returns a reservation object that points to a file on the file system.
2. Reading from the internal system, map the non-repeating parts of the PIP into an RNO in TSpace.

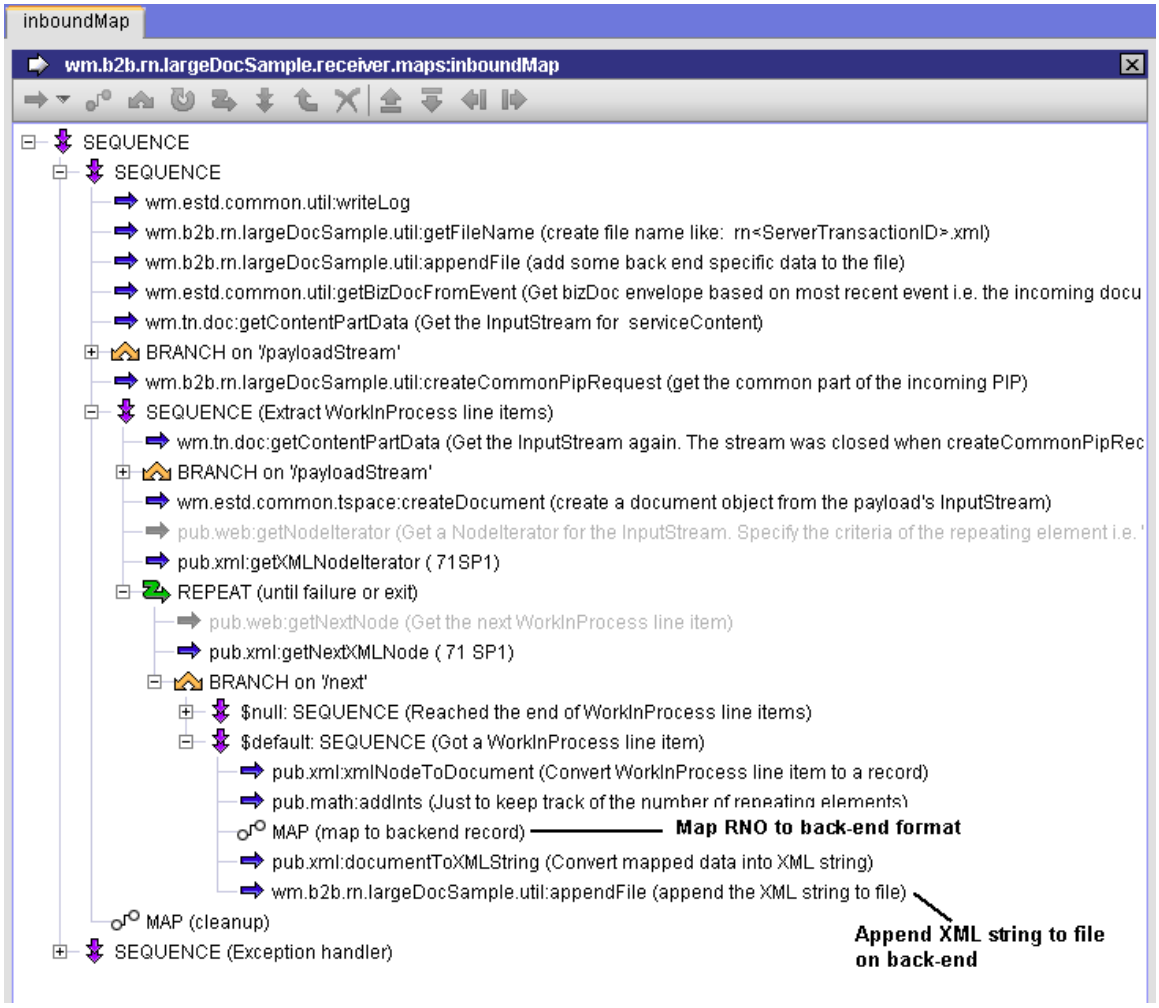
   Note: Write the PIP to TSpace in the order of the elements in the PIP. For example, if the non-repeating parts are at the end of a PIP, write the repeating parts to TSpace, first, and then write the non-repeating parts.

3. Next, invoke the pub.xml:getXMLNodeIterator service to map the repeating parts of the PIP into an RNO in TSpace. This service is used to loop over an array of IData objects. For information about this service, see the Integration Server built-in services reference guide for your release. See "About this Guide" for specific document titles.
4. Write the end tag that corresponds to the root tag of the business document.
5. Close the reservation in TSpace.
6. Invoke the wm.tn.doc:addContentPart service to prevent the reservation object you created from being garbage collected and to persist it to the *BizDocEnvelope* , which is already saved in the Trading Networks database.
7. Map the reservation object to the *documents\Payload IData* object.

Step 5: Customize the Inbound Mapping Service

The inbound mapping service performs the reverse action of the outbound mapping service. In your inbound mapping service, map an RNO to the format used by your internal system.

The following figure illustrates an inbound mapping service from the PIP 3D8 Sample 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 internal object.

Step 6: Customize the Validation Service

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

Important: For large documents, you must specify the ValidateService in the TPA. Otherwise, the PIP is not validated.

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

To create a validation service

1. In Designer, 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`, the validation service has failed.
4. In the TPA for the PIP, enter this validation service name as the **ValidateService** parameter.

   Note: RosettaNet Module invokes the validation service while validating an inbound business document. For an outbound business document, RosettaNet Module only invokes the validation service if the value for the **ValidateOutput** parameter is `yes`.

To see the working example, open the sample PIP 3D8 TPA in My webMethods. The value for **ValidationService** is set as wm.b2b.rn.largeDocSample.validation:validatePip3D8.

What Happens After Customizing and During Run Time?

During run time, your internal systems sends a business document to RosettaNet Module. RosettaNet Module processes the business document, that is, it invokes a Trading Networks service to recognize the TN XML document type, creates the *BizDocEnvelope* , and sends the *BizDocEnvelope* to Process Engine.

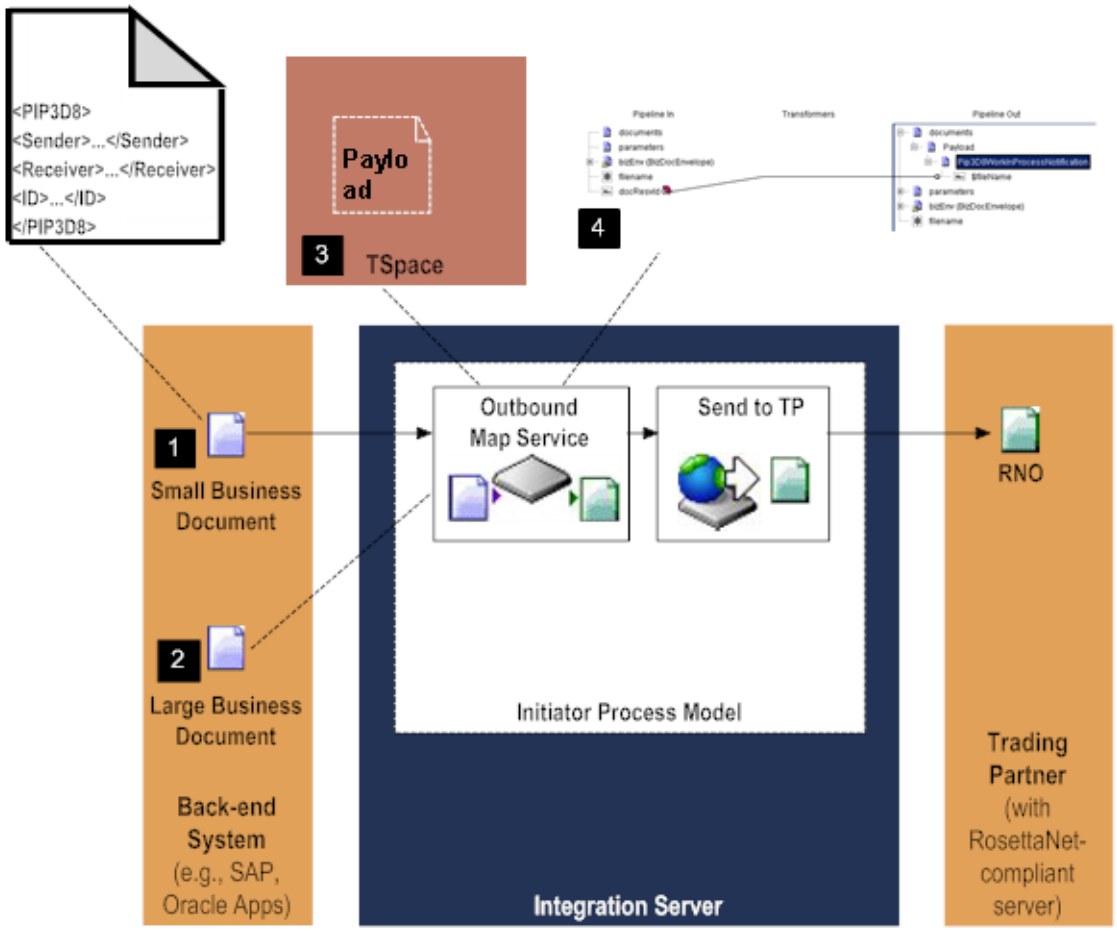
In this scenario, your internal system could send either a small or large business document to RosettaNet Module. For example, in the PIP 3D8 Large Document Sample, the internal system sends a small document that includes a file name that references the large business document.

Alternatively, the internal system could send a large business document. After Trading Networks creates the *BizDocEnvelope* and opens a reservation to TSpace, Trading Networks invokes wm.tn.doc:getContentPart to retrieve the document content and writes the content to TSpace.

The better option of these two methods is to use the small business document to initiate a RosettaNet transaction for a large business document. This improves performance because RosettaNet Module does not need to load the entire large document into memory.

Outbound Mapping of a Large Business Document

Use the following example of a PIP 3D8 Large Document Sample as a model for customizing your outbound mapping service. The following diagram illustrates the process flow for the outbound mapping of a large business document. The table following the diagram explains each step.

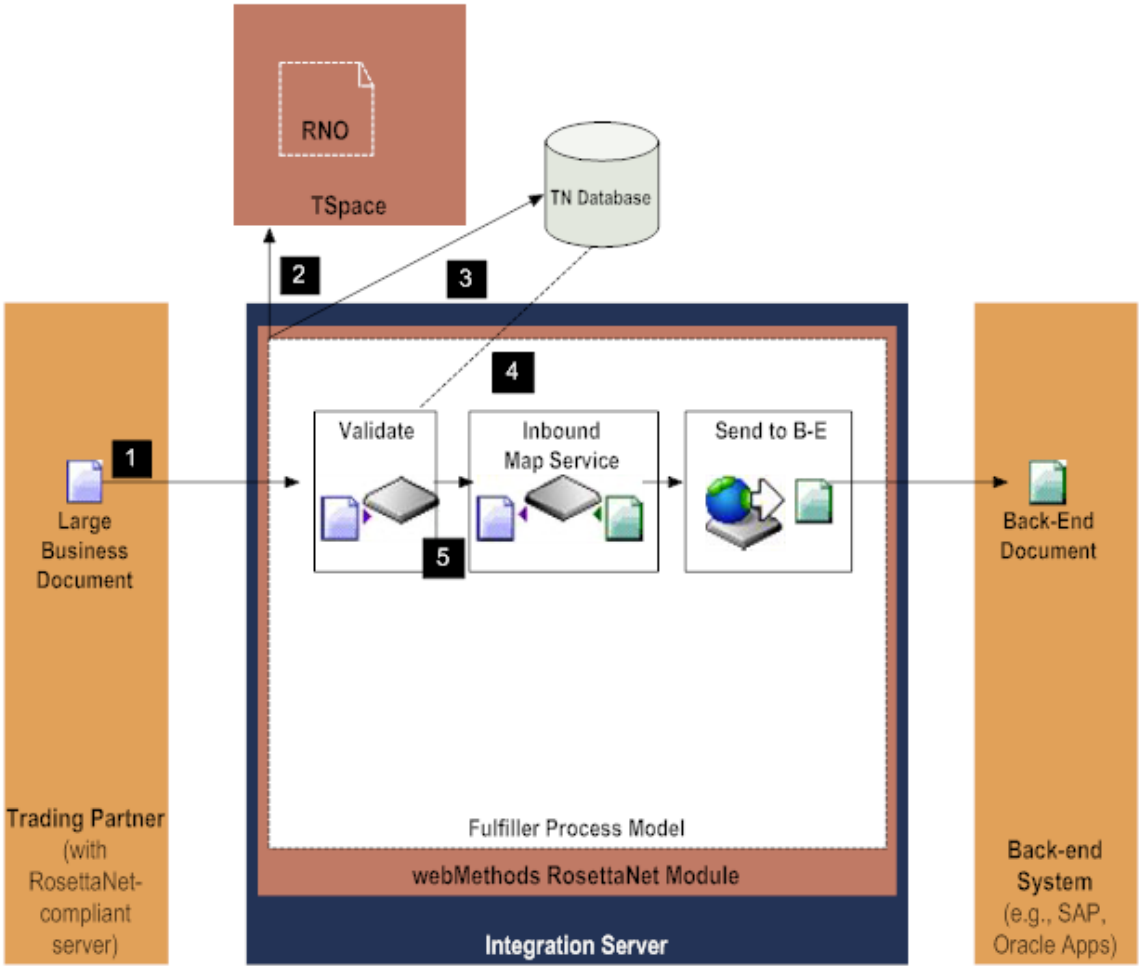


Step	Description
1	The internal system sends an outbound small business document that contains a reference to a large business document to RosettaNet Module. In the Initiator Process Model, the small business document is passed to an outbound mapping step.

Step	Description
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 internal system through a Java InputStream.
3	The outbound mapping service reads each part of the large business document from the internal system, and writes the payload to TSpace. Saving the payload instead of the large document saves valuable memory. After writing the payload to TSpace, the outbound mapping service closes the file.
4	The outbound mapping service then maps the filename presented by *docResvId* into the outgoing business document with the variable name of *$fileName* .

Inbound Validation of a Large Business Document

The following diagram illustrates the inbound validation of a large business document.



Step	Action
1	A trading partner sends a large business document to RosettaNet Module at a fulfiller site. RosettaNet Module recognizes that the business document is a large business document by the value in the *tn.BigDocThreshold* parameter in the Trading Networksproperties.cnf file. For more information about this file, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.
2	RosettaNet Module stores the large business document in TSpace as configured in the watt.server.tspace `location`, `max`, and `timeToLive` parameters.
3	RosettaNet Module saves the large business document in the Trading Networks database.

Step	Action
4	During the conversation, at the **Validation** step, a validation service reads the non-repeating 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 and repeating parts.
5	After validation, the large business document is passed to the Inbound Map Service step, where processing continues.

D Configuration Properties

Controlling RosettaNet Requests in a Processing Queue

RosettaNet Module allows you to specify how many threads can run in parallel when processing RosettaNet documents. By specifying a value in the *estd.queueContainer.noOfThreads* property, you can optimize your document processing.

The *estd.queueContainer.noOfThreads* property allows you to specify the maximum number of threads that should be available in the processing queue for processing requests. Based on the system load, the thread with the least load gets picked from the queue and assigned the processing task. As more transactions arrive for processing, thread selection continues in this fashion.

To specify the maximum number of threads in a processing queue

1. Open the config.cnf file in a text editor from the following directory:

   Integration Server_directory\packages\WmEstdCommonLib\config\

2. Add or edit the following property:

   estd.queueContainer.noOfThreads=n

   where *n* specifies the maximum number of threads available for processing RosettaNet requests. If this property is not defined, the default number of threads is 5.

   The value of this parameter must be less than the Integration Server thread pool size, as specified in the *watt.server.threadPool* parameter. *watt.server.threadPool* defines the maximum number of threads that Integration Server maintains in the server thread pool. Consider the value of this parameter, as well as load conditions, when you define *estd.queueContainer.noOfThreads* . For more information about configuring the server thread pool, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

3. Save and close the configuration file.

4. Restart Integration Server.

Configuring RosettaNet to Archive RNOs

By default, RosettaNet Module archives all business documents (except large business documents) that it sends and receives to the specified archive directory. You can retrieve these archived RNOs by accessing the archive directory and opening the archived RNOs in a text editor.

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 off this level of archiving. You can archive large business documents by configuring the RosettaNet.largeDocument.archive parameter.

To archive inbound or outbound RNOs to the file system

1. Open the config.cnf file in a text editor from the following directory:

   Integration Server_directory\packages\WmRosettaNet\config

2. Change the value of the *RosettaNet.archive.archiveLocation* parameter to the directory where you want to archive your documents. The default directory is *Integration Server_directory* \packages\WmRosettaNet\archive. The archive directories are as follows:

   - <*archivelocation*>\transport\receiving for archived inbound RNOs.
   - <*archivelocation*>\transport\sending for archived outbound RNOs.

   Important: Some Operating Systems allows only specified number of characters for filenames, and this filename length also includes the length of the entire directory structure where the file is stored. As a result of this limitation, in some cases, the archived RNO file name is too long, and you cannot execute any file operations like copy, move, download, retrieve for further recording or storage, on the archived RNO.

   Software AG recommends to keep the default archive location to a directory close to root directory. This is to ensure that, when the Integration Server installation is in a directory that is far from the root directory, the Operating Systems limitation regarding file name lengths does not affect the file operations on archived RNOs.

3. Set the value of the *RosettaNet.archive.required* parameter as true, to archive documents.

   To archive large documents, set the value of *RosettaNet.largeDocument.archive* parameter to true.

4. Set the value of the *RosettaNet.archive.maxArchiveSize* parameter to the maximum directory size for archiving documents. The default value is `-1`, which indicates unlimited storage of archived RNOs. If you exceed the maximum limit set, old archives are deleted to make space for the new archives. The value of the *RosettaNet.archive.maxArchiveSize* parameter is expressed in bytes.

   Note: RNO archive file names are composed of a series of parts, separated by underscores and uses the format: *TNDocType* _*UniqueID* _*TimeStamp* .rno, where

   - *TNDocType* is the Trading Networks document type.
   - *UniqueID* is a part of the ConversationID, that is unique.
   - *TimeStamp* is the time when the RNO is archived.

   For example, an archived RNO might be named as PIP3A4 v1.1 Purchase Order RequestActionDocument_c0a80103f70d4a250000009e1342791898139_1342791899465.rno.

5. Save and close the configuration file.
6. Reload the WmRosettaNet package.

Configuring the Default ACL for a Receive Service

As a security measure, the default ACL for the "pub.estd.rosettaNet:receive" on page 100 service is set to Administrators. You can change the ACL for this service in Integration Server and Designer using the RosettaNet Module parameter *RosettaNet.receive.acl* .

To change the default ACL for the receive service

1. Open the config.cnf file in a text editor from the following directory:

   Integration Server_directory\packages\WmRosettaNet\config
2. Change the value for *RosettaNet.receive.acl* to the desired ACL. For example, *RosettaNet.receive.acl=Anonymous* .
3. Save and close the configuration file.
4. Reload the WmRosettaNet package.

Configuring IS Document Formats

When IS documents are imported from previous versions of IS, the resulting payload may not be in sync with the imported IS document. By specifying values for the IS Document Format configuration properties, you can maintain the order and dimensionality of the elements of the resulting payload.

To configure IS Document Formats

1. Open the config.cnf file in a text editor from the following directory:

   Integration Server_directory\packages\WmRosettaNet\config
2. Update the following properties:
   - *RosettaNet.document.structureFormat* =`true`
   - *RosettaNet.NodeTodocument.structureFormat* =`true`

   Note: If *RosettaNet.document.structureFormat* and *RosettaNet.NodeTodocument.structureFormat* properties are not present in the config.cnf file, add these properties to the config.cnf file and set value as `true`.
3. Save and close the configuration file.
4. Reload the WmRosettaNet package.

Disabling HTML-Encoding for Payload

By default, HTML-encoding is enabled for RosettaNet message processing. This ensures that payloads containing special characters such as `< > , & " '` are also processed correctly.

If your RosettaNet payload data does not contain any special characters, you can disable HTML encoding for successful payload processing.

To disable HTML-Encoding for payload

1. Open the config.cnf file in a text editor from the following directory:

   Integration Server_directory\packages\WmRosettaNet\config

2. Set value of *RosettaNet.document.disableHTMLEncode* as `true`.

   Note: If *RosettaNet.document.disableHTMLEncode* property is not present in the config.cnf file, add the property to the config.cnf file and set value as `true`.

3. Save and close the configuration file.
4. Reload the WmRosettaNet package.

Configuring Delivery Header based on use of secure communication protocol

The delivery header field *isSecureTransportRequired/AffirmationIndicator* indicates whether secure transport is required. The value of *isSecureTransportRequired/AffirmationIndicator* in RNIF 2.0 is determined by the value of the TPA *sign* field by default.

According to some PIP specifications, for example, PIP 3A6 - Purchase Order Status Notification, the value of *isSecureTransportRequired/AffirmationIndicator* can also be determined by use of secure transport protocol (https). If you want to allow value of *isSecureTransportRequired/AffirmationIndicator* field to be determined according to use of secure communication protocol, configure the *RosettaNet.AffirmationIndicator.IsSecureTransportRequired* parameter in the config.cnf file.

To configure delivery header based on secure transport protocol

1. Open the config.cnf file in a text editor from the following directory:

   Integration Server_directory\packages\WmRosettaNet\config

2. Set value of *RosettaNet.AffirmationIndicator.IsSecureTransportRequired* as `true`.

Note: If *RosettaNet.AffirmationIndicator.IsSecureTransportRequired* property is not present in the config.cnf file, add the property to the config.cnf file and set value as `true`.

3. Save and close the configuration file.
4. Reload the WmRosettaNet package.

Configuring *x-RN-Response-Type* header field

The x-RN-Response-Type header field in RNIF 2.0 indicates whether the RosettaNet message is synchronous or asynchronous. The x-RN-Response-Type header field is set with value as `sync` for synchronous messages, and `async` for asynchronous messages.

To remove the x-RN-Response-Type header field from asynchronous messages, configure the *RosettaNet.Headers.SetAsyncHeader* parameter in the config.cnf file.

To configure x-RN-Response-Type header field

1. Open the config.cnf file in a text editor from the following directory:

   Integration Server_directory\packages\WmRosettaNet\config

2. Set value of *RosettaNet.Headers.SetAsyncHeader* as `false`.

   Note: If *RosettaNet.Headers.SetAsyncHeader* property is not present in the config.cnf file, add the property to the config.cnf file and set value as `true`.

3. Save and close the configuration file.
4. Reload the WmRosettaNet package.

E Managing RosettaNet Cache

Overview

webMethods RosettaNet Module internally uses caching to store information related to each RosettaNet conversation. This cached information is used for processing RosettaNet conversations. The RosettaNet Module internal cache is same as the cache type used with the Integration Server.

If you are using a clustered environment, the internal RosettaNet Module cache also shares the data among different nodes of the cluster which helps in processing a transaction for a conversation on a node different from where it originally started.

For information on caching and clustering used with the Integration Server, see the Integration Server administration guide and the Integration Server clustering guide for your release. See "About this Guide" for specific document titles.

Managing the RosettaNet Module Cache

The name of the RosettaNet Module cache is *RNModelSessionCache*. By default, the entries in the *RNModelSessionCache* are not deleted by the RosettaNet module, because the RosettaNet module cannot determine the duration of the running process or PIP. As a result, the *RNModelSessionCache* size increases with each conversation.

Depending upon the type of cache used, that is, Coherence or Ehcache, the steps for clearing the cache differs.

Clearing the *RNModelSessionCache* when using Coherence

When Coherence is the cache type associated with Integration Server, you can clear the *RNModelSessionCache* by doing one of the following:

- Restarting Integration Server.
- Configuring the time, in minutes, after which the cache must be cleared.

The below procedure explains how to set the time after which, the *RNModelSessionCache* must be cleared.

To configure time after which cache must be cleared - with cache type Coherence

1. Shut down the Integration Server.
2. Open the < *Integration Server_directory* >\config\Caching\webm-cache-config.xml file.
3. Add the following tags in the webm-cache-config.xml file.

If you are using...	Add the following to your webm-cache-config.xml file...
Single node environment	<localCache> <name>RNModelSessionCache</name> <localScheme> <expiryDelay>*value*</expiryDelay> <flushDelay>*value*</flushDelay> </localScheme> </localCache>
Clustered environment	<replicatedCache> <name>RNModelSessionCache</name> <replicatedScheme> <backMap> <localScheme> <expiryDelay>*value*</expiryDelay> <flushDelay>*value*</flushDelay> </localScheme> </backMap> </replicatedScheme> </replicatedCache> **Note:** If you are using Integration Server Version 7.1.3, in the <name> tag, add IS Cluster:<*the name of the cluster*>:RNModelSessionCache. For example, if the name of the cluster is *Seller_Cluster*, the <name> tag must be as follows: <name>IS Cluster:Seller_Cluster:RNModelSessionCache</name>

The following step contains details about the XML tags and their values.

4. The XML tag values you must specify are as follows:

For this XML tag...	Specify...
<expiryDelay>	The time after which the contents of the cache is considered as expired. The contents of the cache can be deleted only after it is marked for expiration. Use the format: <expiryDelay>*value*</expiryDelay>, where *value* is the time after which the contents are marked as expired. Example: <expiryDelay>15m</expiryDelay>, where 15m indicates 15 minutes of time. Value can be given in the following formats: ■ MS or ms (milliseconds) ■ S or s (seconds) ■ M or m (minutes) ■ H or h (hours)

For this XML tag...	Specify...
	■ D or d (days)
<flushDelay>	The time interval between periodic cache flushes of the expired cache content. Use the format: <flushDelay>*value*</flushDelay>, where *value* is the time interval. Example: <flushDelay>20m</flushDelay>, where 20m indicates 20 minutes of time. That is, the cache flushes the expired content every 20 minutes. Value can be given in the following formats: ■ MS or ms (milliseconds) ■ S or s (seconds) ■ M or m (minutes) ■ H or h (hours) ■ D or d (days)

Note: Set the delay values as greater than the expiration time of the process models of the existing PIPs being used. If the delay values are set wrongly, the objects that are needed later, may get cleaned up. This will result in errors during transaction.

5. Restart the Integration Server.

Clearing the *RNModelSessionCache* when using Ehcache

To clear the *RNModelSessionCache* when Ehcache is the type of cache associated with Integration Server, see the Integration Server administration guide for your release. See "About this Guide" for specific document titles.

Index

V

W