SeeBeyond ICAN Suite

RosettaNet ETD Library User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Version 20050406031727.

Contents

Chapter 1

Introduction

This chapter introduces you to the RosettaNet ETD Library User's Guide.

1.1 Overview

An Event Type Definition (ETD) library contains sets of pre-built structures for industry-standard formats. e*Gate ETD files are message format definitions in two formats:

- Monk
- Java

The RosettaNet ETD Library is a feature of the e*Gate Integrator system, and contains message definitions for RosettaNet messages. This document gives a brief overview of RosettaNet and the RosettaNet message structures provided with e*Gate, and provides information on installing and using the RosettaNet ETD Library.

1.2 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the e*Gate system, to have familiarity with Windows and/or UNIX operations and administration, and to be thoroughly familiar with Windows-style GUI operations.

1.3 Compatible Systems

The e*Gate RosettaNet ETD Library is available on the following platforms:

- Windows 2000, Windows XP, and Windows Server 2003
- HP-UX 11.0 and 11i (PA-RISC)
- IBM AIX 5.1L
- Sun Solaris 8

- Japanese Windows 2000, Windows XP, and Windows Server 2003
- Japanese HP-UX 11.0 and 11i (PA-RISC)
- Japanese Sun Solaris 8

Note: ***UNIX Systems****—This guide uses the backslash ("\") as the separator within path names. If you are working on a UNIX system, make the appropriate substitutions.*

Chapter 2

RosettaNet Overview

This chapter presents an overview of RosettaNet, including the structure of an RosettaNet envelope and its data elements.

2.1 RosettaNet Overview

2.1.1. What Is RosettaNet?

RosettaNet is an organization set up by leading information technology companies to define and implement a common set of standards for eBusiness.

RosettaNet was among the first to recognize the eBusiness model for supply chain automation—namely, that companies must share processes as well as transactions in order to truly automate the supply chain. By defining a standard set of processes that could be implemented by any company, RosettaNet transcends the EDI model to fully automate the buying and selling process.

RosettaNet aims to align the business processes of supply chain partners. This goal is achieved by the creation of Partner Interface Processes or PIPs. Each PIP defines how two specific processes, running in two different partners' organizations, will be standardized and interfaced across the entire supply chain. A PIP includes all business logic, message flow, and message contents to enable alignment of the two processes. This facilitates electronic exchange of standard business documents between trading partners. A PIP is a complete business transaction.

RosettaNet is the only popular standard currently using business process models.

RosettaNet PIPs also identify the security components associated with the business process.

RosettaNet is used within the US and also internationally.

RosettaNet message flows are all interactive; messages are not batched.

2.2 Components of the RosettaNet Standard

The RosettaNet standard utilizes the following components:

- Framework or enveloping layers
- Partner Interface Process (PIP) specifications or processes
- Dictionaries or vocabulary
- IT and Electronic Components (EC)
- Industry-standard data model with 450 classes for different kinds of parts, each with attributes to specify that kind of part
- XML syntax
- Internet transport

2.3 Message Guideline Format

Specification of RosettaNet message guidelines is in human-readable form, using RTF and HTML formats. Additionally, message guidelines are provided in machine-readable formats. The preferred format is XML DTDs. Message vocabulary comes from RosettaNet dictionaries, and each message guideline has its own DTD or schema.

Although these DTDs allow partners to determine if a message structure is valid, they do not allow partners to determine if a message is valid with respect to a message guideline for a business document (captured in the RosettaNet business document UML model). This is because neither DTDs nor XML schema languages are as rich as the UML and OCL (Object Constraint Language) that RosettaNet uses to describe business documents designed during PIP analysis sessions. Because of this, the only complete specification of a message guideline is in the human-readable RTF and HTML formats.

2.4 What Is a Message Structure?

The term *message structure* refers to the way in which data elements are organized and related to each other for a particular eBusiness message. Each message structure consists of the following:

- Physical hierarchy

  The predefined way in which envelopes, segments, and data elements are organized to describe a particular RosettaNet eBusiness message.

- Delimiters

  The specific predefined characters that are used to mark the beginning and end of envelopes, segments, and data elements. In RosettaNet, the XML begin and end tags are the delimiters.

- Properties

  Characteristics of a data element, such as the length of each element, default values, and indicators that specify attributes of a data element—for example, whether it is required, optional, or repeating.

  The properties of a RosettaNet XML tag are defined by the DTD.

e*Xchange Partner Manager uses e*Gate Event Type Definitions, which are based on the RosettaNet message structures, to interpret the inbound or outbound messages. e*Gate installation includes message structures for a number of RosettaNet eBusiness messages.

2.5 Supported Versions

The RosettaNet ETD Library supports the versions of the RosettaNet Implementation Framework (RNIF) listed below.

- Monk message structures:
  - RNIF 1.1
  - RNIF 2.0
- Java message structures:
  - RNIF 2.0

2.6 Structure of a RosettaNet Business Message

The RosettaNet Service protocol message is the business message that is exchanged between two RosettaNet entities (Services and/or Agents).

This message is made up of several parts, all in XML format, packaged into a MIME multipart/related content-type message.

The basic message structure varies between RNIF 1.1 and RNIF 2.0.

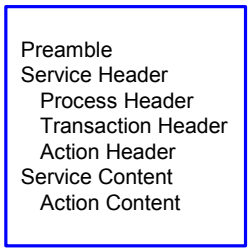
2.6.1. RNIF 1.1 Structure

An RNIF 1.1 message has the following structure:

- Preamble (administrative XML DTD)
- Service Header (administrative XML DTD) consisting of three parts:
  - Process Header
  - Transaction Header
  - Action Header
- Service Content (message-specific DTD) including Action Content

Figure 1 shows this basic structure.

Figure 1 Structure of an RNIF 1.1 Business Message



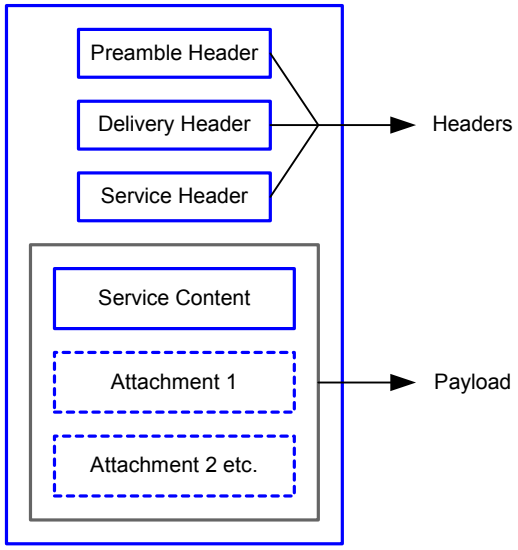
2.6.2. RNIF 2.0 Structure

An RNIF 2.0 message has the following structure:

- Preamble (administrative XML DTD)
- Delivery Header (administrative XML DTD)
- Service Header (administrative XML DTD)
- Service Content (message-specific DTD)
- Optionally, one or more attachments

Figure 2 shows this basic structure.

Figure 2 Structure of an RNIF 2.0 Business Message



2.6.3. Preamble

All RosettaNet messages must have a Preamble. It is specified with a DTD that is common across all messages.

The Preamble section of the MIME message contains elements that are global to the RosettaNet service and others that are common to the Service Header and Service Content; for example, the message standard and version used in the message.

An example of an actual RNIF 1.1 Preamble is shown in Figure 3.

Figure 3 Example of an RNIF 1.1 Preamble

```
<!DOCTYPE Preamble SYSTEM
"PreamblePartMessageGuideline.dtd" >
<Preamble>
<VersionIdentifier>1.1</VersionIdentifier>
<DateTimeStamp>19990531T132000.0500Z</
DateTimeStamp>
<GlobalAdministeringAuthorityCode>RosettaNet
</GlobalAdministeringAuthorityCode>
<GlobalUsageCode>Test</GlobalUsageCode>
</Preamble>
```

2.6.4. Delivery Header (RNIF 2.0 Only)

The delivery header is new in RNIF 2.0, and facilitates the routing of messages through hubs. It contains the following attributes:

- Elements for the sending and receiving of trading partner identities
- Message date and time stamp

- Globally unique tracking ID

Every RNIF 2.0 business message has a Delivery Header. The information in it is used by everyone involved in routing the message, including sender, receiver, and any relay point (such as a hub).

Even if the message is encrypted, the Delivery Header is never encrypted.

2.6.5. Service Header

The Service Header is specified with a DTD that is common across all messages. A separate DTD and/or XML schema for each message validates the body of the messages.

The Service Header helps identify the following items:

- The PIP
- The activity and the action to which the message belongs
- The PIP instance
- Information about the sender of the message
- Information about the recipient of the message
- The date and time at which the message was sent
- Whether the message is a Test message or a Production message
- Whether the sender is to be treated as an "unknown partner"

The Service Header format is fixed and independent of the specifics of the message contained in the payload. However, the Service Content could change based on variance in the business content.

2.6.6. Service Content (Payload)

Service Content is specified in individual PIPs. Each PIP has one or more business actions that are described by means of individual DTDs or schema.

The payload part of a RosettaNet Business Message—that is, the actual business content—is a message in XML format.

The RosettaNet Service Content is the same in both RNIF 1.1 and RNIF 2.0, except that in RNIF 2.0 it can contain non-RosettaNet content.

2.6.7. Attachments (RNIF 2.0 Only)

An RNIF 2.0 Action Message can contain one or more attachments. These are often supporting documents accompanying the business documents. Each attachment must have the MIME Content-ID attribute specified.

Attachments can be in any of many formats, including the following:

- Microsoft Word documents
- GIF or TIF graphic images

- PDF files

2.6.8. Differences Between 1.1 and 2.0

Some of the important differences between RNIF 1.1 and 2.0 are listed below.

- Multiple transfer protocols—RNIF 1.1 supported HTTP and HTTPS; RNIF 2.0 also supports SMTP.
- Attachments—RNIF 2.0 allows the capability to add attachments to the message.
- Encryption of Service Content and Service Header—Encryption via S/MIME is a new feature in RNIF 2.0.
- Synchronous Transactions—RNIF 2.0 allows synchronous exchange of request and response messages within the same HTTP session, for certain PIPs.
- Digital Signatures—RNIF 2.0 uses S/MIME for signing RosettaNet business messages.
- Message Manifest—RNIF 2.0 supports a new Message Manifest that describes the contents of the payload.
- Changes to Service Header—The RNIF 2.0 Service Header includes new information and capabilities.
- Signals—RNIF 2.0 no longer supports the Acceptance Acknowledgment Signal.

2.6.9. Backward Compatibility

RNIF 2.0 is not backward compatible with RNIF 1.1. Although e*Xchange Partner Manager supports both versions, they are not interchangeable.

However, PIPs published prior to the release of RNIF 2.0—specifically, the version 1.0 and 1.1 PIPs— should work with RNIF 2.0. PIPs published after RNIF 2.0 might work with 1.1, but RosettaNet does not guarantee this.

2.7 RosettaNet eBusiness Message Structures

This section provides general information on RosettaNet eBusiness messages.

2.7.1. Clusters

RosettaNet has delineated the scope of supply chain processes for which it will design PIPs. This scope is divided into a total of 17 segments grouped in 6 clusters. The clusters are listed in Table 1.

Table 1 RosettaNet Clusters

Cluster Name	Type of Process
Cluster 1	Partner, Product, and Service Review
Cluster 2	Product Introduction
Cluster 3	Order Management
Cluster 4	Inventory Management
Cluster 5	Marketing Information Management
Cluster 6	Service and Support

2.7.2. Partner Interface Processes (PIPs)

In RosettaNet, a PIP is a model that depicts the activities, decisions and partner Role Interactions that fulfill a business transaction between two partners in a supply chain.

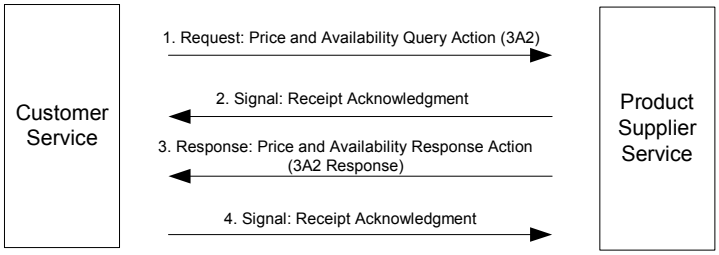
Each partner participating in the partner interface process must fulfill the obligations specified in a PIP instance. If any one party fails to perform a service as specified in the PIP implementation guide then the business transaction is null and void.

A PIP is a complete eBusiness activity; it is a business process subset. It commonly includes at least two separate messages and often more.

The messages involved in a PIP business document exchange can be classified into two broad categories—business action messages and business signal messages.

An example of the message exchange that might occur for a specific PIP, 3A2, is shown in Figure 4.

Figure 4 Pricing/Availability Request & Response: RosettaNet



2.7.3. RosettaNet Enveloping Features

RosettaNet provides the following enveloping features:

- Always interactive and acknowledged
- You can configure the following for each message:
  - Timeouts and retries
  - Optional non-repudiation (audit of message and signature)

- Optional authorization (certificates)

2.8 Types of RosettaNet Messages

There are two types of Service Content (Payload) messages in RosettaNet:

- Action message
- Signal message

The Preamble, Service Header, and Service Content in RNIF 1.1, and Preamble, Delivery Header, Service Header, and Service Content in RNIF 2.0, are all XML documents which need to be validated against their RosettaNet-provided DTDs.

The Preamble and the Service Header have DTDs that are common to all RosettaNet Service Messages and are defined and explained by RosettaNet's Preamble Part Message Guideline and Service Header Part Message Guideline respectively.

The Service Header itself comprises elements for the Process Header, Transaction Header, and Action Header.

The Delivery Header (RNIF 2.0 only) is defined and explained in the Delivery Header Message Guideline.

The Service Content varies based on the actual business message being exchanged (as defined in individual PIPs) with each business message that can be a Service Content defined by a RosettaNet guideline of its own.

Action Messages

The action message is the fundamental business message that is exchanged within a transaction of a RosettaNet Business Process and is specified in corresponding PIPs.

Business Signals

A RosettaNet business signal is a system-level positive or negative acknowledgment message or exception message that might be returned to either the sender or receiver of a PIP.

There are five types of business signals:

- Receipt Acknowledgment

  A Receipt Acknowledgment is a positive signal acknowledging receipt of a Business Action message. It is sent when a message is received by a trading partner and is found to be a structurally and syntactically valid RosettaNet business action message. It is sent only if required by the PIP (almost all PIPs require a receipt acknowledgment).

- Receipt Acknowledgment Exception

  A Receipt Acknowledgment Exception is a negative signal indicating a problem with a Business Action message.

- Acceptance Acknowledgment (not supported in RNIF 2.0)

  An Acceptance Acknowledgment is a positive signal acknowledging acceptance of a business document for processing. It is sent when the Service Content is valid with respect to the business rules of the receiving trading partner and can be accepted for processing by the trading partner. It is sent only if required by the PIP.

- Acceptance Acknowledgment Exception (not supported in RNIF 2.0)

  An Acceptance Acknowledgment Exception is a signal representing a negative acknowledgment of acceptance due to business rule processing errors.

- General Exception

  A General Exception is a negative signal indicating a processing exception.

Note: *Business Actions are acknowledged. Business Signals are never acknowledged.*

2.9 About XML

RosettaNet uses XML (eXtensible Markup Language), which has the following features:

- Begin/end tag delimited syntax
- Tags are usually nested into related fields or data objects
- Metadata allows bilateral change between trading partners
- Web browsers now parse XML documents
- Tags are user-defined or industry-defined
- DTDs usually accompany XML documents, to specify what tags to expect

2.10 Security

RosettaNet includes the following security features:

- Transport via HTTP with SSL (HTTPS)
- RosettaNet "business object" which contains:
  - Digital signature length (4 bytes)
  - Optional digital signature (PKCS#7): sender's certificate & public key
- Digital certificates—both ends authenticated:
  - SSL requires server to have a certificate
  - RosettaNet additionally requires client to have a certificate
- Optional non-repudiation:
  - An audit trail of the message and signature

- The PIPs specify which messages should use non-repudiation

- RNIF 2.0 only: Encryption of Service Content and Service Header via S/MIME

2.10.1. Use of MIME Within RosettaNet

RosettaNet business messages consist of a Service Header and a message body. Both the header and the body are complete, valid XML documents. The header and the body are encoded within a multipart/Related MIME message.

2.10.2. Use of Digital Certificates Within RosettaNet

Digital certificates are delivered as part of the application/PKCS7-signature part of a multipart-signed RNIF 1.1 or 2.0 message.

RosettaNet requires that the sender include any certificates that contain the signer's public key. The sender can also, if desired, include the associated issuer certificates.

2.10.3. Encryption

Encryption is a new feature offered with RNIF 2.0. In e*Xchange, the level of encryption is set in the message profile. There are three choices:

- 0—No encryption
- 1—encryption of Service Content and any attachments
- 2—encryption of Service Header, Service Content, and any attachments

The preamble and delivery header are never encrypted, since they contain information necessary to the correct routing of the message.

Encryption of Inbound Messages

For an inbound message, e*Xchange uses the encryption setting from the partner profile to check if the message has been encrypted according to the setting. If the message does not have the correct portions encrypted, e*Xchange goes into the error handling process.

For more information on how e*Xchange handles errors with RosettaNet messages, refer to the *e*Xchange Partner Manager User's Guide*.

Encryption of Outbound Messages

For an outbound message, e*Xchange encrypts the message according to the encryption setting specified in the partner profile.

2.10.4. Digital Signatures

RosettaNet versions 1.1 and 2.0 approach digital signatures a little differently:

- In RNIF 1.1, the signature is in the RosettaNet Object format (raw) at the end of the message.
- RNIF 2.0 uses S/MIME for signing RosettaNet business messages. The signature is in Base64, separated by S/MIME headers at the end of the message if there are no attachments.

2.10.5. SSL Keystores

RNIF 1.1 and 2.0 support the use of SSL Keystores for HTTPS communication.

In the e*Xchange Partner Manager, SSL keystores are supported only if you are using the Web interface. The keystore is not included in the RosettaNet object (RNIF 1.1) or business message (RNIF 2.0) but instead is included as part of the standard Event going out of e*Xchange for the Java HTTPS e*Way.

2.11 Implementation

2.11.1. e*Gate Support of RosettaNet

e*Gate includes some RosettaNet Event Type Definitions (ETDs) pre-built from RosettaNet DTDs in two versions: Java and Monk.

e*Xchange Partner Manager (e*Xchange) supports the RosettaNet enveloping methodology. The various parameters can be configured in the e*Xchange user interface.

Note: *The e*Xchange Partner Manager only supports the Monk version of the RosettaNet ETD Library templates. However, the Java RosettaNet templates can be used outside e*Xchange but within e*Gate. For example, a Java template file from the RosettaNet ETD Library can be used for mapping an inbound message to the Standard Event (also in Java), and the resulting Standard Event sent to e*Xchange Partner Manager.*

Structures

In e*Gate, a number of ETDs for all RosettaNet versions are pre-built and are part of the RosettaNet structure libraries. To customize a structure or build a new one, use the e*Gate Java or Monk ETD Editor.

To create a new e*Gate ETD from a RosettaNet DTD, the XML Toolkit must be installed. The e*Gate ETD Editor invokes the XML Toolkit during conversion.

If you are creating a Monk ETD, choose the XML Converter within the build option for the Monk ETD Editor.

If you are creating a Java ETD, use the DTD Wizard within the Java ETD Editor (from the **File** menu, choose **New**).

Validation

RosettaNet supports a logical segmentation of validation steps. e*Gate performs the validation on each part of the RosettaNet Business Message.

In RosettaNet, incoming messages go through the following validation steps:

- Grammar Validation
- Sequence Validation
- Schema Validation
- Content Validation

Grammar and sequence validations are performed against the Service Header DTD. Service Content validation is not done until after schema validation. If you are using the e*Xchange Partner Manager, note that e*Xchange does not validate service content.

Note: *The Service Header might be valid according to the DTD even if the actual Service Content contains errors.*

Translations

e*Gate does not contain any pre-built translations for RosettaNet. You can build these scripts in the e*Gate Collaboration Rules Editor GUI, which provides a user-friendly drag-and-drop front end for creating Monk scripts.

Acknowledgments

RosettaNet Business Signal Receipt Acknowledgments and Receipt Acknowledgment Exceptions are automatically handled by the e*Xchange Partner Manager.

Note: *In e*Xchange Partner Manager, you should never select a RosettaNet Receipt Ack Exception as the expected return message.*

2.11.2. Implementation Guides

RosettaNet PIPs can be used as implementation guides. RosettaNet publishes one PIP per multi-step process.

Each PIP document includes:

- A UML activity diagram for multi-transaction process
- Specification for how to implement digital conversation (acknowledgments, wait times, retries, and digital signatures)
- One DTD for each of the multiple messages

All PIPs include the following three views of the business model:

- Business Operational View (BOV), the high-level business view. This includes:
  - **UML** Activity Diagram
  - Table of roles in the process
  - Table of activities (nodes) in the process
- Functional Services View (FSV). The functional services view specifies the network component services and agents and the interactions necessary to execute PIPs.
- Implementation Framework View (IFV). The implementation framework view specifies the network protocol message formats and communications requirements that need to be in place for the exchange of messages to occur.

2.11.3. Trading Partner Agreements

It is normal for trading partners to have individual agreements in addition to the standard RosettaNet PIPs. The electronic messages have to meet standards on two levels:

- Compliance with RosettaNet requirements.
- Compliance with any requirements specific to a trading partner.

2.12 Additional Information

For more information on RosettaNet, visit **www.rosettanet.org**.

Note: *This information is correct at the time of going to press; however, SeeBeyond has no control over this site. If you find the link is no longer correct, use a search engine to search for "RosettaNet."*

Chapter 3

RosettaNet Template Installation

This chapter provides information on installation of the RosettaNet templates, and the files and folders created as a result of installation.

3.1 Installation Procedure

This section explains how to install the RosettaNet template files.

Note: *This procedure only covers the specific portions of installation that relate to template installation. For complete installation instructions, refer to the* ***e*Gate Integrator Installation Guide****.*

Before you begin:

- You must have Administrator privileges to install back-end components such as the RosettaNet templates.
- Exit all Windows programs, including any anti-virus applications, before running the setup program.
- Verify your e*Gate registry host name, schema name, control broker logical name, and the administrator user name and password.

To install the RosettaNet templates on Windows

1. Log in on the workstation on which you want to install the RosettaNet templates.
2. Insert the installation CD into the CD-ROM drive.

   If Autorun is enabled, the setup program automatically starts. Otherwise:

   - On the task bar, click the **Start** button, and then click **Run**.
   - In the **Open** field, type **D:\setup\setup.exe** where **D:** is your CD-ROM drive.
3. Follow the installation instructions until you come to the **Please choose the product to install** dialog box.
4. Select **e*Gate Integrator**, and then click **Next**.
5. Follow the on-screen instructions until you come to the second **Please choose the product to install** dialog box.
6. Select **Add-ons**, and then click **Next**.

7 Follow the on-screen instructions until you come to the **Select Components** dialog box.

8 Highlight (but do not check) **ETD Libraries**, and then click the **Change** button.

The **Select Sub-components** dialog box appears.

9 Select **RosettaNet ETD Library 4.5.3**.

10 Click **Continue** to return to the **Select Components** dialog box, and then click **Next**.

11 Follow the rest of the on-screen instructions to install the RosettaNet templates.

Note: *Do not change the default directory location for the RosettaNet template files.*

To install the RosettaNet templates on UNIX

1 Follow the steps for the standard e*Gate installation.

For more information, refer to the *e*Gate Integrator Installation Guide*.

2 At the prompt **Choose e*Gate Add-on Application**, enter the number assigned for the RosettaNet ETD Library 4.5.3 (scroll down the list to check).

3 Enter the installation path, or press Enter to accept the default path (recommended).

4 Enter the hostname of the registry server (UNIX host).

The library is installed.

5 Conditional: if you need to install more than one library, repeat steps 2 through 4 as needed.

3.2 Creating a Monk Validation Collaboration for RosettaNet

You can create or customize a Monk validation Collaboration for validating your RosettaNet messages. Create a **.tsc** file that uses the RosettaNet message structures provided in the RosettaNet ETD Library.

Note: *If you are using e*Xchange Partner Manager, only Monk ETDs and Collaborations are supported. However, you can use Java ETDs and Collaborations to perform validation outside e*Xchange.*

Make sure that the name of the validation function matches the name of the file that contains the function.

Note: *For general information about building the validation rules, refer to the RosettaNet Implementation Framework specification for the RosettaNet version you are using.*

To create a Monk validation Collaboration Rules Script for RosettaNet

1 Using the e*Gate Collaboration Editor, create a Collaboration Rules Script.

For RosettaNet 1.1, you must place the Collaboration Rules Script in or below the **monk_scripts\common** folder; for example:

<*eGate*>\Server\registry\repository\default\monk_scripts\common\etc.

If you are using e*Xchange, place them in this path:

<*eGate*>\Server\registry\repository\eXSchema\monk_scripts\common\etc.

For RosettaNet2.0, it is recommended but not required that you place the Collaboration in this location:

<*eGate*>\Server\registry\repository\default\monk_scripts\common\RNIF2.0

If you are using e*Xchange, you can place the Collaborations in any location as long as the location is defined in the "additional monk path" in the configuration of the eX_ePM e*Way.

2 In e*Xchange, in the message profile for the transaction, in the **General** section, specify the name of the Collaboration.

3 In the Schema Designer, in the configuration parameters for the eX_ePM e*Way, add the path to the Collaboration file to the "Additional Monk Path" parameter.

After invoking the validation routine, the main process function checks a global Monk variable, "error_data," for any error conditions that failed the validation process. Thus, any error detected in the validation routine should be indicated by setting this global Monk variable. The user can use the utility function (util-add-to-error) for this purpose. For example, to indicate an invalid component ID, you could use the following code segment:

```
    ...
    (if (component_ID_is_valid? ...) ;;;; component_ID_is_valid? can
be a user defined function that validates the component ID.
                (begin
                )
                (begin
                   (set! error_data (util-add-to-error error_data
"3010^Invalid Component ID"))
                )
            )
    ...
```

Note: *An error component consists of an error code and an error string, separated by the "^" character. Use error codes 3000–4999 for user-defined error conditions.*

3.3 Creating a Java Validation Collaboration

If you want to create a Java validation Collaboration, use the Java version of the e*Gate Collaboration Editor to create the Collaboration Rules Script.

Since the RosettaNet ETD Library does not support RNIF 1.1, you would be creating a validation Collaboration for the version 2.0 ETDs. The Collaboration can be stored in any location.

For more information on using the e*Gate Collaboration Editor, refer to the *e*Gate User's Guide*.

3.4 Reinstalling the Library

If you are reinstalling a later version of the RosettaNet ETD Library, it is a good idea to first manually delete your existing library, and then reinstall. This will ensure that any legacy files that are no longer needed are deleted.

However, if you have customized your ETD files, do not delete your existing files.

Chapter 4

The RosettaNet ETD Library

This chapter provides information on the files and folders that comprise the RosettaNet ETD Library.

There are two sets of files provided, to accommodate the two languages supported by the e*Gate Event Type Definition editor:

- Java

  The Java library supports RNIF 2.0 only.

- Monk

  The Monk library supports both major versions of the RosettaNet Implementation Framework (RNIF 1.1 and 2.0), and PIP versions 1.0, 1.1, 1.2, 1.3, 1.4, and 2.0.

4.1 PIPs Provided With the Library

The RosettaNet business message version 1.0 and 1.1 PIPs (Monk) that are provided with the RosettaNet ETD Library are listed in Table 2. In some cases, there are multiple versions to support multiple Service Content versions.

Table 2 RNIF 1.1 PIPs (Monk only)

PIP Name	Service Content Version				
	1.0	1.1	1.2	1.3	1.4
▪ PIP0A1 - Failure Notification	Yes				
Product Review					
♦ PIP1B1 - Manage Product Subscription	Yes				
Prepare for Distribution					
♦ PIP2A1 - Distribute New Product Information	Yes				
♦ PIP2A2 - Query Product Information	Yes				
♦ PIP2A5 - Query Technical Information	Yes				
♦ PIP2A8 - Distribute Stock Keeping Unit (SKU)	Yes	Yes			
Quote and Order Entry					
♦ PIP3A2 - Query Price and Availability		Yes	Yes	Yes	

Table 2 RNIF 1.1 PIPs (Monk only) (Continued)

PIP Name	Service Content Version				
	1.0	1.1	1.2	1.3	1.4
♦ PIP3A3 - Transfer Shopping Cart	Yes	Yes			
♦ PIP3A4 - Manage Purchase Order	Yes	Yes	Yes	Yes	Yes
♦ PIP3A5 - Query Order Status	Yes	Yes			
♦ PIP3A6 - Purchase Order Status	Yes	Yes			
♦ PIP3A7 - Notify of Purchase Order Acceptance		Yes	Yes	Yes	

The RosettaNet business message version 2.0 PIPs (Monk) that are provided with the RosettaNet ETD Library are listed in Table 3. In some cases, there are multiple versions to support multiple Service Content versions.

Table 3 RNIF 2.0 PIPs (Monk Library)

PIP Name	1.0	1.1	1.2	1.3	1.4	2.0
ROS20_Exception_20.ssc						Yes
ROS20_RecAck_20.ssc						Yes
▪ PIP0A1 - Failure Notification						Yes
Product Review						
♦ PIP1B1 - Manage Product Subscription	Yes					
Prepare for Distribution						
♦ PIP2A1 - Distribute New Product Information	Yes					
♦ PIP2A2 - Query Product Information	Yes					
♦ PIP2A5 - Query Technical Information	Yes					
♦ PIP2A8 - Distribute Product Stock Keeping Unit (SKU)	Yes	Yes				
♦ PIP2A9 - Query Electronic Component Technical Information	Yes					
Quote and Order Entry						
♦ PIP3A2 - Query Price and Availability		Yes	Yes	Yes		
♦ PIP3A3 - Transfer Shopping Cart	Yes	Yes				
♦ PIP3A4 - Manage Purchase Order	Yes	Yes	Yes	Yes	Yes	
♦ PIP3A5 - Query Order Status	Yes	Yes				
♦ PIP3A6 - Distribute Order Status	Yes	Yes				
♦ PIP3A7 - Notify of Purchase Order Acceptance		Yes	Yes	Yes		
♦ PIP3B1 - Distribute Transportation Projection	Yes					
♦ PIP3B2 - Notify of Advance Shipment	Yes					

Table 3 RNIF 2.0 PIPs (Monk Library) (Continued)

PIP Name	1.0	1.1	1.2	1.3	1.4	2.0
• PIP3B3 - Distribute Shipment Status	Yes					
• PIP3B4 - Query Shipment Status	Yes					
• PIP3B5 - Request Shipment Change	Yes					
• PIP3B6 - Notify of Shipments Tendered	Yes					
• PIP3C2 - Request Financing Approval	Yes					
• PIP3C3 - Notify of Invoice	Yes					
• PIP3C4 - Notify of Invoice Reject	Yes					
• PIP3C5 - Notify of BIlling Statement	Yes					
• PIP3C6 - Notify of Remittance Advice	Yes					
• PIP3D8 - Distribute Work in Process	Yes					
• PIP3D9 - Query Work in Process	Yes					
• PIP4A4 - Notify of Planning Release Forecast	Yes					
• PIP4B2 - Notify of Shipment Receipt	Yes					
• PIP4C1 - Distribute Inventory Report	Yes					
• PIP5D1 - Request Ship from Stock and Debit Authorization	Yes					
• PIP5D2 - Notify of Blanket Ship from Stock and Debit Authorization	Yes					

The RosettaNet business message version 2.0 PIPs (Java) that are provided with the RosettaNet ETD Library are listed in Table 4.

Table 4 RNIF 2.0 Supported PIPs (Java Library)

PIP Name	Version
Segment 1A: Partner Review	
• PIP1A1 - Request Account Setup	vB01.00.00A
• PIP1A2 - Maintain Account	vB01.00.00B
• PIP1B1 - Manage Product Information Subscriptions	v1.0
Segment 2A: Prepare for Distribution	
• PIP2A1 - Distribute New Product Information	v1.0
• PIP2A2 - Query Product Information	v1.0
• PIP2A3 - Query Marketing Information	vB01.00.00A
• PIP2A4 - Query Sales Promotion & Rebate Information	vB01.00.00A
• PIP2A5 - Query Technical Information	v1.0
• PIP2A6 - Query Product Lifecycle Information	vB01.00.00A
• PIP2A7 - Query Product Discontinuation Information	vB01.00.00A
• PIP2A8 - Distribute Product Stock Keeping Unit (SKU)	v1.1

Table 4 RNIF 2.0 Supported PIPs (Java Library) (Continued)

PIP Name	Version
♦ PIP2A12 - Distribute Product Master	R01.00.00
Segment 2B: Product Change Notification	
♦ PIP2B1 - Change Basic Product Information	vB01.00.00B
♦ PIP2B2 - Change Marketing Information	vB01.00.00A
♦ PIP2B3 - Change Sales Promotion & Rebate Information	vB01.00.00A
♦ PIP2B4 - Change Product Technical Information	vB01.00.00A
♦ PIP2B5 - Change Product Lifecycle Information	vB01.00.00A
Segment 2C: Product Design Information	
♦ PIP2C1 - Distribute Engineering Change Status	R01.00.00
♦ PIP2C2 - Request Engineering Change	R01.00.00
♦ PIP2C3 - Distribute Engineering Change Response	R01.00.00
♦ PIP2C4 - Request Engineering Change Approval	R01.00.00
♦ PIP2C5 - Notify of Engineering Change Order	R01.00.00
♦ PIP2C6 - Notify of Engineering Change Implementation Plan	R01.00.00
Segment 3A: Quote and Order Entry	
♦ PIP3A1 - Request Quote	vR02.00.00
♦ PIP3A2 - Query Price and Availability	R02.00.00A
♦ PIP3A3 - Transfer Shopping Cart	vR02.00.00
♦ PIP3A4 - Manage Purchase Order	vR02.00.00
♦ PIP3A5 - Query Order Status	vR02.00.00
♦ PIP3A6 - Distribute Order Status	vR02.00.00
♦ PIP3A7 - Notify of Purchase Order Acceptance	vR02.00.00
♦ PIP3A8 - Request Purchase Order Change	vR01.00.00
♦ PIP3A9 - Request Purchase Order Cancellation	vR01.00.00
♦ PIP3A10 - Notify of Quote Acknowledgment	vR01.00.00
Segment 3B: Transportation and Distribution	
♦ PIP3B1 - Distribute Transportation Projection	R01.00.00
♦ PIP3B2 - Notify of Advance Shipment	R01.01.00
♦ PIP3B3 - Distribute Shipment Status	v01.00.00
♦ PIP3B4 - Query Shipment Status	R01.00.00
♦ PIP3B5 - Request Shipment Change	R01.00.00
♦ PIP3B6 - Notify of Shipments Tendered	R01.00.00
Segment 3C: Returns and Finance	
♦ PIP3C2 - Request Financing Approval	R01.00.00
♦ PIP3C3 - Notify of Invoice	R01.00.00

Table 4 RNIF 2.0 Supported PIPs (Java Library) (Continued)

PIP Name	Version
• PIP3C4 - Notify of Invoice Reject	R01.00.00
• PIP3C5 - Notify of BIlling Statement	R01.00.00
• PIP3C6 - Notify of Remittance Advice	R01.00.00
Segment 3D: Product Configuration	
• PIP3D8 - Distribute Work in Process	vR01.00.00
• PIP3D9 - Query Work in Process	vR01.00.00
Segment 4A: Collaborative Forecasting	
• PIP4A1 - Notify of Strategic Forecast	R01.00.00A
• PIP4A2 - Notify of Embedded Release Forecast	R01.00.00A
• PIP4A3 - Notify of Threshold Release Forecast	R01.00.00A
• PIP4A4 - Notify of Planning Release Forecast	R02.00.00A
• PIP4A5 - Frequently Asked Questions	R01.00.00A
Segment 4B: Inventory Allocation	
• PIP4B2 - Notify of Shipment Receipt	v1.0
Segment 4C: Inventory Reporting	
• PIP4C1 - Distribute Inventory Report	R02.00.00
Segment 5C: Design Win Management (Electronic Components)	
• PIP5C1 - Distribute Product List	v01.00.00
• PIP5C2 - Request Design Registration	vB01.00.00G
• PIP5C3 - Create/Request Design Win	vB01.00.00C
• PIP5C4 - Distribute Registration Status	vB01.00.00F
• PIP5C5 - Query Registration Status	vB01.00.00D
Segment 5D: Ship from Stock and Debit (Electronic Components)	
• PIP5D1 - Request Ship from Stock and Debit Authorization	vR01.00.00
• PIP5D2 - Notify of Blanket Ship from Stock and Debit Authorization	vR01.00.00
PIP0A1 - Notification of Failure	v1.0

4.2 Folder Structure Created by Installation

By default, installation places the RosettaNet templates in the locations shown in Table 5.

Table 5 RosettaNet Template Locations

These files…	Are installed in this location…
Java	\<eGate>\Server\Registry\Repository\default\ETD\templates\RosettaNet\ros20\RNBMLib
Monk	\<eGate>\Server\Registry\Repository\default\monk_scripts\templates\RosettaNet

Installation commits the templates to the **default** schema on the Registry Host that you specified during the installation process.

Within the Monk template directory, there is a subfolder for each RNIF version. Java ETDs are provided for RosettaNet 2.0 only; however, these are still stored in a \ros20 subfolder.

Each version-specific folder has a subfolder for each Service Content version. For each version, there is an .**ssc** file for each Monk RosettaNet message structure, and an .**xsc** and a .**jar** file for each Java RosettaNet message structure.

Figure 5 shows a sample directory structure for the RosettaNet Monk templates.

Figure 5 Folder Structure for RosettaNet Monk Templates

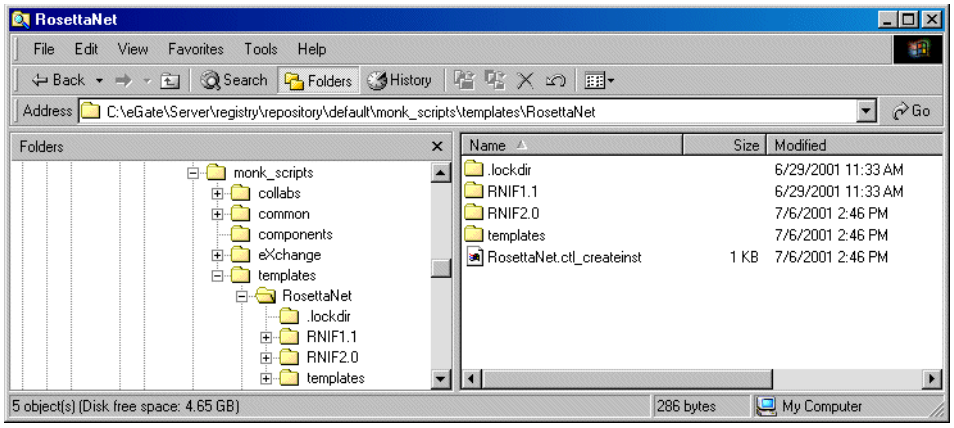
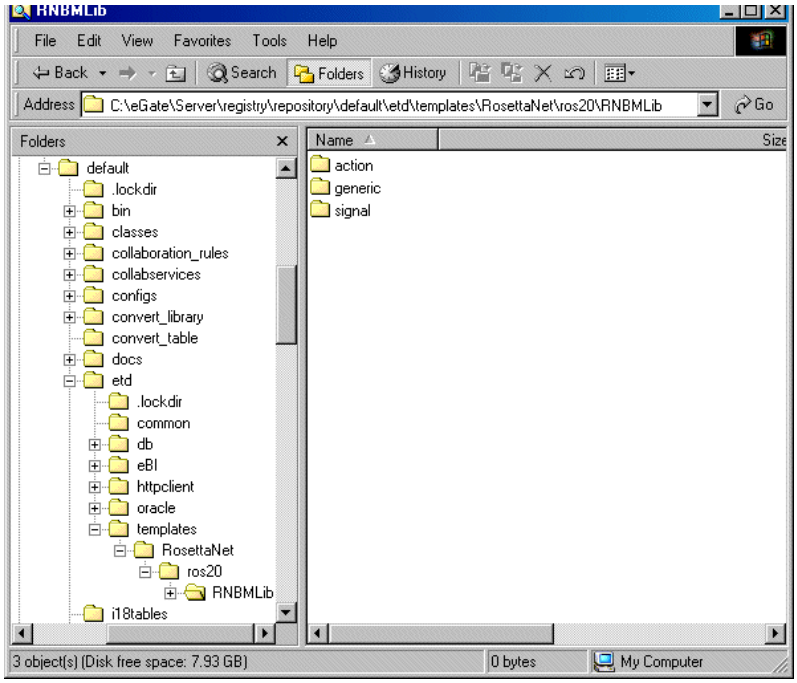


Figure 6 shows a sample directory structure for the RosettaNet Java templates.

Figure 6 Folder Structure for RosettaNet Java Templates



Note: *There are several versions of the Service Content. Some PIPs are available in all versions; others are only available for certain versions. For example, 0A1 and 2A1 were only released in version 1.0. For a full list of PIPs provided, see* **"PIPs Provided With the Library" on page 26**.

UNIX installation places the files in the same path locations and directories as shown above.

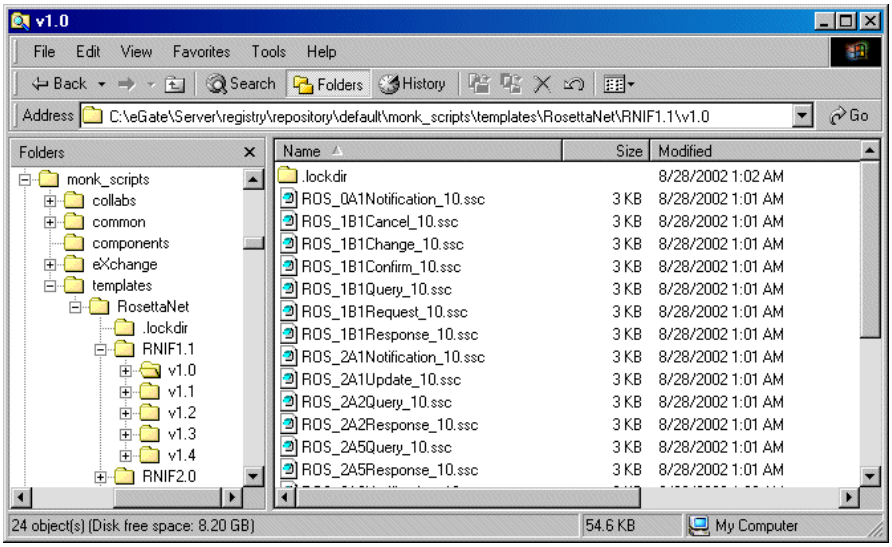
4.3 RosettaNet Files

Installation of RosettaNet includes a number of files, needed by e*Gate for processing the various supported PIPs.

4.3.1. Monk Files

Figure 7 shows the template files for RNIF 1.1, Service Content version 1.0.

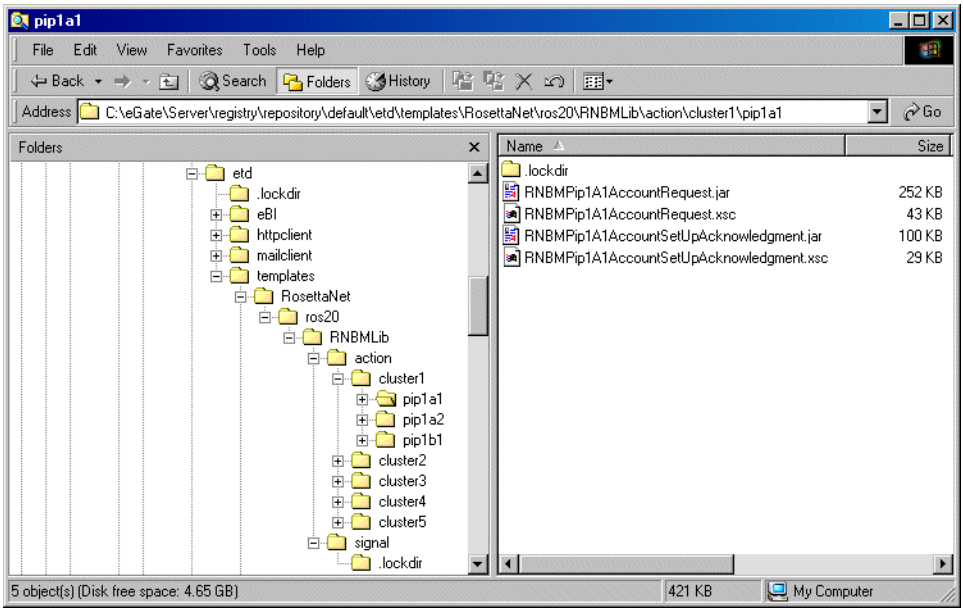
Figure 7 RNIF 1.1 Version 1.0 Monk Template Files



4.3.2. Java Files

Figure 8 shows the template files for the RNIF 2.0, Service Content version 2.0, Java files that are in the \ros20 subfolder.

Figure 8 RNIF 2.0 Java Template Files



4.4 RosettaNet Template Files

The file name for each template is comprised of the same set of elements in the same sequence. The file name for a business signal has a different structure than that of a business message.

Breakdown and examples are given below.

4.4.1. Business Signals—Monk

The file names for business signals in RNIF 1.1 are constructed as follows:

- BS_ (to indicate a business signal)
- Abbreviated name of the business signal
- _nn to indicate the version number; for example, _11 for RNIF 1.1
- .ssc (file extension indicating an e*Gate message structure)

The e*Gate RosettaNet library includes **.ssc** files for all the RosettaNet business signals listed in Table 6.

Table 6 RosettaNet Business Signals in the ETD Library (Monk, RNIF 1.1)

Business Signal	File Name
Acceptance Acknowledgment	BS_AcceptAck_11.ssc
Acceptance Acknowledgment Exception	BS_AcceptAckExcept_11.ssc
General Exception	BS_Exception_11.ssc
Receipt Acknowledgment	BS_RecAck_11.ssc
Receipt Acknowledgment Exception	BS_RecAckExcept_11.ssc

4.4.2. Business Messages—Monk

The file names for Monk business messages are constructed as follows:

- ROS_ (for RNIF 1.1) or ROS20_ (for RNIF 2.0)
- Name of message (for example, 2A8 or 3A4)
- Abbreviation for message type (for example, Accept or Cancel)
- Identification of version, one of the following:
  - _10 for version 1.0
  - _11 for version 1.1
  - _12 for version 1.2
  - _13 for version 1.3
  - _14 for version 1.4
  - _20 for version 2.0

- .ssc (file extension)

A few examples are provided in Table 7.

Table 7 RosettaNet Business Messages in the ETD Library: Sample Files

Business Action Message	File Name
RNIF 1.1 Version 1.0 Failure Notification	ROS_0A1Notification_10.ssc
RNIF 1.1 Version 1.3 Purchase Order Acceptance	ROS_3A4Accept_13.ssc
RNIF 2.0 Version 1.1 Purchase Order Cancellation (Cancel)	ROS20_3A4Cancel_11.ssc
RNIF 2.0 Version 1.4 Purchase Order Request	ROS20_3A4Request_14.ssc

4.4.3. Business Messages—Java

The file names for Java business messages are constructed as follows:

- RNBM (for RosettaNet Business Message)
- Pip
- Name of message (for example, 2A8 or 3A4)
- Abbreviation for message type (for example, Accept or Cancel)
- .xsc (file extension)

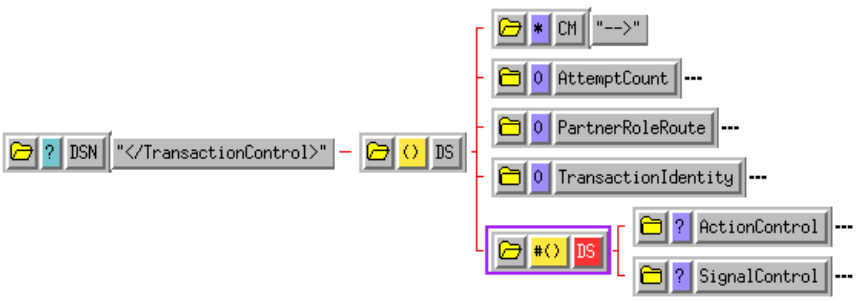
For example: **RNBMPip1A1AccountRequest.xsc**

4.5 Using the RosettaNet ETD Library to Build Translations

There is an important point to bear in mind when using RosettaNet ETDs to build e*Gate message translations.

As part of the basic functionality of XML, there might be a situation where a required parent node might have two or more optional child nodes. For example, in the portion of **ROS_3A6Notification_10.ssc** shown in Figure 9, the parent node **Transaction Control** is required and the child nodes **Action Control** and **Signal Control** are optional.

Figure 9 Required Parent Node with Optional Child Nodes



Although both child nodes are optional, when writing the translation it is vital that you provide a value for either one or the other. If you do not write to one of the child nodes, the e*Gate message structure will generate invalid XML.

The node shown in Figure 9 is common to all the RosettaNet ETDs.

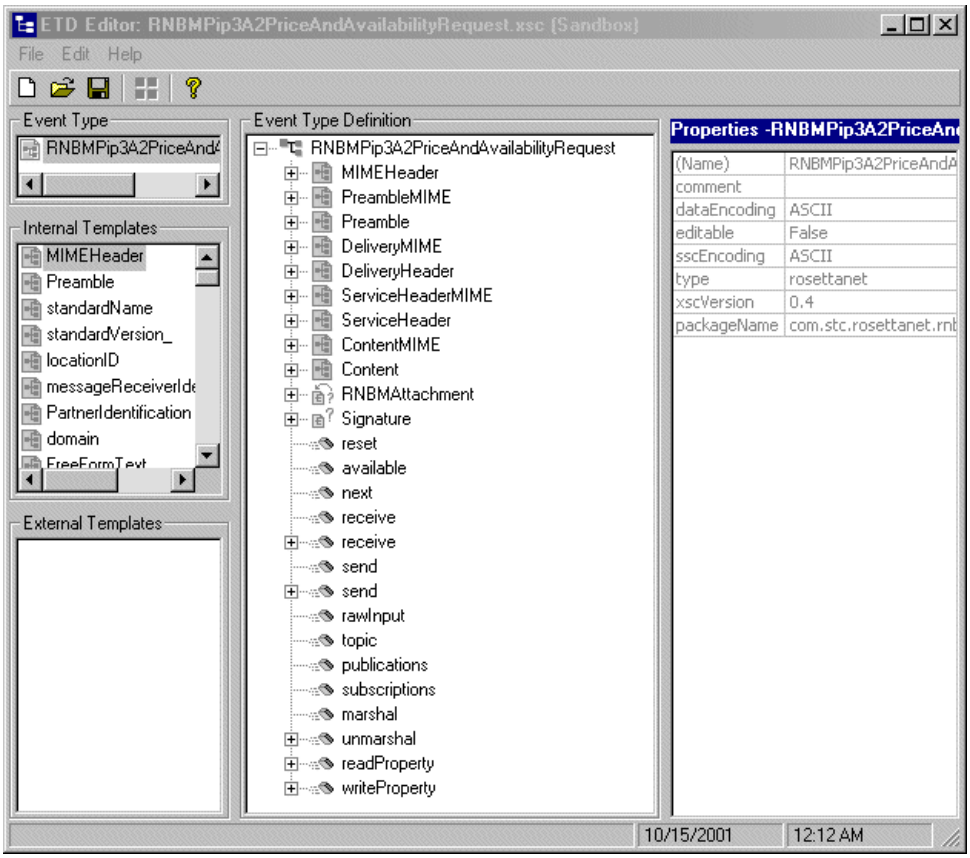
This is a basic XML principle, and therefore applies to all the XML structures you create. Whenever there is a structure where one parent is required and the children are optional, you must provide a value so that e*Gate knows which child node to populate.

4.6 Viewing a RosettaNet ETD in the Java ETD Editor

You can open up a RosettaNet ETD in the Java ETD Editor. However, you cannot edit the ETD.

Figure 10 shows PIP 3A2, Price and Availability Request, in the Java ETD Editor.

Figure 10 PIP 3A2, Price and Availability Request, in the Java ETD Editor



4.7 Customizing a Java ETD

Currently SeeBeyond does not support the editing of pre-built Java ETDs. However, e*Gate offers a feature that allows you to convert existing Monk ETDs (**.ssc** files) to Java-enabled ETDs (**.xsc** files). This feature is the SSC Wizard.

To create a customized Java ETD

1. Create a corresponding Monk ETD, or use the Monk version of the Java ETD if available.
2. Customize the Monk ETD (**.ssc** file) using the e*Gate ETD Editor.
3. Convert the Monk ETD to a Java ETD using the e*Gate SSC Wizard.

   When the conversion is done, you have three files:

   - The original Monk ETD (**.ssc** file)
     Keep this file in case further customization is needed.
   - The Java version of the ETD (**.xsc** file)
   - The corresponding **.jar** file

If you need to make further changes to the ETD, make the changes in the **.ssc** file and run the conversion again.

For specific instructions on using the e*Gate ETD Editor or the SSC Wizard, refer to the *e*Gate Integrator User's Guide*.

Glossary of RosettaNet Terms

business message

The individual business documents involved in a PIP (action and signal messages) are exchanged in a container that packs together other related entities such as headers and digital signatures. This container with its constituent parts is the basic unit of exchange between two RosettaNet end-points, and is known as a "RosettaNet Business Message."

A RosettaNet Business Message always contains a Preamble header, a Service Header, and a Service Content.

business object

A "business object" is a message plus a digital signature, version number, and length.

Business Operational View (BOV)

All PIPs include three views of the business model. One of these is the Business Operational View. It is the high-level business view.

cluster

RosettaNet has delineated the scope of supply chain processes for which it will design PIPs. This scope is divided into a total of 17 segments grouped in 6 clusters. The clusters and segments serve as a mechanism to group all supply chain processes into a manageable framework.

CA

An acronymn for Certification Authority. A Certification Authority is an application that puts a public key into a certificate and digitally signs it to guarantee that the public key belongs to the stated owner.

CRL

An acronymn for Certificate Revocation List. A CRL is a binary file that allows clients and servers to check whether the entity to be verified has a current, valid digital certificate. Normally, the Certification Authority (CA) is responsible for the non-repudiation of transactions, for maintaining an audit log, and for caching CRLs.

Document Type Definition (DTD)

DTDs usually accompany XML documents, to specify what tags to expect.

All DTDs for RosettaNet messages, including those which are not specific to an individual PIP, are issued as both Message Guidelines (HTML document) and DTDs. These are part of the RosettaNet Implementation Framework, although documented separately.

DTD
See Document Type Definition.

Functional Service View (FSV)
All PIPs include three views of the business model. One of these is the Functional Service View. It is the technical document that specifies the network component services and agents and the interactions necessary to execute PIPs.

guideline
A set or collection of specifications, sometimes including specific implementation advice.

header
Control information prepended to the content.

Implementation Framework View (IFV)
All PIPs include three views of the business model. One of these is the Implementation Framework View. It is a technical document, more detailed than the FSV, and specifies the network protocol message formats and communications requirements that need to be in place for the exchange of messages to occur.

Partner Interface Process (PIP):
A model that depicts the activities, decisions and partner Role Interactions that fulfill a business transaction between two partners in a given supply chain. Each partner participating in the partner interface process must fulfill the obligations specified in a PIP instance. If any one party fails to perform a service as specified in the PIP implementation guide then the business transaction is null and void.

payload
In RosettaNet 2.0, the payload of a business message includes the Service Content plus any file attachments.

payload container
The Service Header and Service Content.

PIP
See Partner Interface Process (PIP)

Preamble
The Preamble section of a RosettaNet message contains elements that are global to the RosettaNet service and those that are common to the Service Header and Service Content. It is specified with a DTD that is common across all messages (**preamble.dtd**).

Service Content
The "Service Content" of a RosettaNet message is the business message.

Service Header
The Service Header is specified with a DTD that is common across all messages. A separate DTD and/or XML schema for each message validates the body of the messages.

The Service Header format is fixed and independent of the specifics of the message contained in the payload. However, the Service Content might change based on variance in the business content.

standard

A set of clearly defined and agreed-upon conventions for specific programming interfaces that has been approved by a formally constituted standards-setting body. RosettaNet has created a standard for eBusiness messages.

validation

The verifying of all or part of a message against the requirements of the specification. Validation of a RosettaNet message compares the message against the content and sequence requirements for the RosettaNet message standard. A data element, action, transaction, or process is valid only if it meets all requirements of the RNIF specification, as well as all requirements of the relevant PIP specification.

XML document

A data object written in XML (eXtensible Markup Language). An XML document is made up of virtual storage units called entities, containing either parsed or unparsed data. Parsed data is made up of characters, some of which form the character data in the document and some of which form markup. Markup encodes a description of the document's storage layout and logical structure.

Index

R

S

T

U

V

W

X