RosettaNet Implementation Framework: Core Specification

Version: Validated 02.00.00

13 July 2001

This color indicates changes made for version R02.00.00A
This color indicates changes made for version R02.00.00B
This color indicates changes made for version R02.00.00C
This color indicates changes made for version V02.00.00

Insignificant spelling or formatting changes are not highlighted.

Legal Disclaimer

RosettaNet™, its members, officers, directors, employees, or agents shall not be liable for any injury, loss, damages, financial or otherwise, arising from, related to, or caused by the use of this document or the specifications herein, as well as associated guidelines and schemas. The use of said specifications shall constitute your express consent to the foregoing exculpation.

Copyright

Trademarks

RosettaNet, Partner Interface Process, PIP and the RosettaNet logo are trademarks or registered trademarks of "RosettaNet," a non-profit organization. All other product names and company logos mentioned herein are the trademarks of their respective owners. In the best effort, all terms mentioned in this document that are known to be trademarks or registered trademarks have been appropriately recognized in the first occurrence of the term.

Additional Disclaimers

Inclusion of Document Type Definitions (DTDs) and Element descriptions in this document are for ease of comprehension. While every effort has been made to ensure that what appears in this document matches the separately published DTD files (*.dtd) and Message Guideline specifications associated with this document, in the event of discrepancies, the DTD file or Message guideline specification is to be used.

Use of examples throughout this document is intended to illustrate the concepts or rules being discussed. They must not be treated as specifications themselves.

Contents

Figures

Tables

Examples

Version History

Version 01.00	8 June 1999	Release.
Version 01.00.01	8 July 1999	Release.
Version 01.01.00	30 December 1999	Release.
Release 02.00.00	3 January 2001	Release for Validation.
Release 02.00.00A	25 April 2001	Validation Update (Limited Distribution).
Release 02.00.00B	07 May 2001	Validation Update (Limited Distribution).
Release 02.00.00C	25 June 2001	Validation Update (Limited Distribution).
Validated 02.00.00	11 July 2001	Validated Specification.

Preface

Purpose of the Document

This document is designed to assist e-business system implementers and solution providers who wish to create or implement interoperable software application components that cooperatively execute RosettaNet PIPs. The document does this by specifying the exchange protocol that enables participating supply chain members to implement RosettaNet PIPs.

The result of these specifications should be to enable two RosettaNet objectives:

- Streamline Execution: RosettaNet needs to facilitate the rapid development of Partner Interface Processes (PIPs).
- Accelerate Adoption: RosettaNet needs to facilitate the rapid development of e-business applications that execute RosettaNet-compliant PIPs.

Intended Audience

1. The primary audience for this document is software engineers who will be developing RosettaNet-compliant networked software applications that can interoperate with RosettaNet-compliant networked software applications developed by other companies. These applications will cooperatively execute RosettaNet e-business PIPs.
2. The secondary audience is system architects, including:
    a. Those within implementing companies who must integrate their architectures with RosettaNet architectures and applications; and
    b. Those who volunteer to participate in RosettaNet projects to create additional RosettaNet e-business specifications.

Prerequisites

RosettaNet assumes that the audience will be familiar with or have knowledge of the following:

- General Internet protocols,
- MIME and S/MIME,
- Digital signatures and the Secure Socket Layer (SSL),
- Extensible Markup Language (XML),
- BNF grammar specification syntax,

- All the external references listed in "References."

Scope of the Document

The focus of this document is specification of the core of the RosettaNet Implementation Framework; that is, packaging, routing, and transferring of RosettaNet business messages (including security aspects), as well as specification of business signal messages used in the execution of RosettaNet Partner Interface Processes or PIPs.

While it provides sufficient business and technical background to understand the context for the implementation framework, the actual specification of the implementation framework core is the focus of this document.

This document does **not** provide either user documentation or a detailed architectural treatise. This document subsumes previous versions, including Technical Advisories that pertained to previous versions.

Structure of This Document

This document is an implementation specification for the RosettaNet networked application architecture. It contains the following sections:

- Section 1, "Introduction" has two parts:
  - "Business Background" introduces new business concepts that provided requirements or otherwise influenced the development of this version of the implementation framework.
  - "Technical Background" introduces new technical concepts that influenced the development of this version of the implementation framework.
- Section 2, "Technical Specifications" has six parts:
  - "RosettaNet Business Message Components" enables the implementer to understand what is needed to populate the various parts of the RosettaNet Business Message.
  - "Security Provisions and Trading Partner Authentication" specifies the use of S/MIME and establishes norms for use of digital signatures.
  - "RosettaNet Business Message Packaging and Unpackaging" specifies how the implementer assembles the defined message components and how the recipient extracts those components.
  - "RosettaNet Business Message Transfer" specifies transport or transfer protocols for RosettaNet Business Message exchange, and specifies which are mandatory and which are optional; additionally, it provides debug header specifications for use in certain situations.

- "Business Signal Specifications & Process Control PIPs" identifies and specifies current business signals, as well as PIPs that are used in controlling the process of PIP business exchanges.
- "Flow of RosettaNet Business Messages" specifies the role of business action messages and business signals in the choreography of a PIP.

- There are several appendices:
  - Appendix A, "Key Differences between RNIF 1.1 & RNIF 2.0" outlines features that are either new in RNIF 2.0 or that have been substantially changed from RNIF 1.1.
  - Appendix B, "Required PIP Metamodel Changes" identifies the changes that are expected to the existing PIP metamodel in order to take full advantage of features added in RNIF 2.0.
  - Appendix C, "IFV Mapping from BOV and FSV" serves to remove "boilerplate" material from the individual PIP specifications and place it in the RNIF.
  - Appendix D, "Importance of Transfer Independence" supports the rationale for transport independence via several example scenarios.
  - Appendix E, "Anticipated Futures" describes some promising technologies that may be useful in future versions of the RNIF.
  - Appendix F, "Additional Examples" offers more extensive examples of PIP exchanges via the RNIF than are present in the specification sections.
  - Appendix G, "References" presents both RosettaNet and other documents that are cited in this document.
  - Appendix H, "Glossary" gives definitions for key words used in this document.

~~Note that, although it is not a part of this document at this time, a separate "Compliance Checklist," consisting of the compliance statements from each subsection of section 2 and arranged in useful and meaningful fashion, will be published shortly after the approval of this document.~~

Use of Normative Specifications

The RosettaNet Implementation Framework specification incorporates by reference certain normative standards or specifications from non-RosettaNet sources. These documents are referenced in the text and are listed in the "References" appendix of this document.

This document does not restate material from the referenced document unless this document is changing a part of the referenced document. The reader is expected to refer to the relevant original source document for the text of referred specifications.

Style Conventions

This specification uses a number of conventions to convey specific meanings. These fall into three categories: typographical conventions, language conventions, and graphical conventions. They are identified below.

Typographical Conventions

The use of a `monospaced font` indicates presentation of a code fragment.

Within the `monospaced font`, the use of *`italics`* indicates that the text so presented is text to be replaced by the user or the system, depending upon the context of the code fragment.

Note: In sections 2.3 (“RosettaNet Business Message Packaging and Unpackaging”) and 2.4 (“RosettaNet Business Message Transfer”), the MIME convention of using angle brackets (“<>”) within the monospaced font to enclose text that is to be replaced has been followed. In these sections, no XML code (which uses angle brackets differently) is presented.

Language Conventions

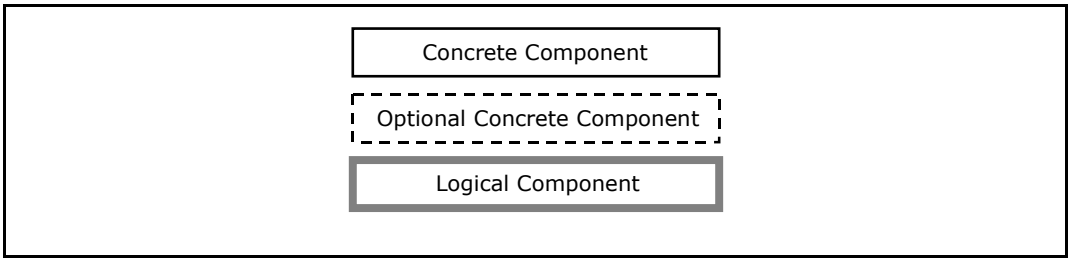
This specification adopts the conventions expressed in the Internet Engineering Task Force’s (IETF) Request for Comments (RFC) 2119 “Key Words for Use in RFCs to Indicate Requirement Levels.” The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in section 2 of this document are to be interpreted as described in RFC 2119.

Formatting Conventions

Examples are used throughout the document to enhance understanding. Therefore, they are formatted for readability. This may mean that lines breaks and extra white spaces have been used in some examples.

Graphical Conventions

Figures that show the message components, as well as the packaging and unpackaging of those components, use various line types to indicate whether something is a concrete component (thin black outline) or a logical component (thick grey line). If a component or packaging method is optional, the line is broken instead of solid.



Backward Compatibility

The following are statements on backward compatibility between RNIF 1.1 and RNIF 2.0:

1. RNIF 2.0 is *not* backward compatible with RNIF 1.1. That is, RNIF 2.0 is not simply a compatible superset of RNIF 1.1. Software solutions that implement *only* RNIF 2.0 WILL NOT be interoperable with software solutions that implement *only* RNIF 1.1 and vice versa.

   If a software solution that implements only RNIF 2.0 receives an RNIF 1.1 message, then the solution is not expected to do anything with that message. It MAY simply choose to ignore that message.

   Subsequent releases of RNIF 2.x will be backward compatible with previous releases of RNIF 2.x. That is, RNIF 2.1 will be backward compatible with RNIF 2.0, as will RNIF 2.2, 2.3, etc.

2. All PIPs published *prior* to the publication of RNIF 2.0 MUST work with RNIF 1.1 and SHOULD work with RNIF 2.x.

3. PIPs published *after* the publication of RNIF 2.0 MUST work with RNIF 2.x and MAY work with RNIF 1.1.

4. RosettaNet will issue a separate communication regarding its “retirement” policy for obsolete releases.

1 Introduction

RosettaNet's mission is to facilitate electronic exchange of standard business documents between trading partners, adhering to the Partner Interface Processes (PIPs) specified and standardized by RosettaNet. Fundamental to this are the RosettaNet Implementation Framework (RNIF), the PIP specifications, and the business and technical dictionaries. This document supplies the specification for the RosettaNet Implementation Framework; separate documents provide PIP and dictionary specifications.

This introductory section provides both business and technical background information that is intended to help the reader make full use of the actual specifications contained in section 2 of this document.

1.1 Business Background

Since the publication of version 1 (and its revisions) of the RosettaNet Implementation Framework (RNIF), changes have occurred both in the way that RosettaNet sees the structure of the framework and in the e-business environment in which RosettaNet members find themselves. This section touches upon those changes and gives the business rationale for certain changes that have been made to the RNIF specifications. See also the "Technical Background" sub-section for additional influences on these specifications.

1.1.1 Implementation Framework Concept

In previous versions of the implementation framework specifications, the subject matter has been limited to specifying the format and elements of the common parts of PIP messages (e.g., headers); and the packaging, routing, and transport of all PIP messages and business signals. It has also included security to a limited extent.

RosettaNet has since realized that this is only a portion of a useful framework that members would need to create robust implementations. Some additional elements of a robust framework would include Trading Partner Agreements and directories or registries.

This document, therefore, covers only a portion of the total RosettaNet Implementation Framework – although it is a very large and important part. Figure 1 shows the relationship of the Implementation Framework and its constituent parts to the rest of the RosettaNet specifications in a trading partner implementation.

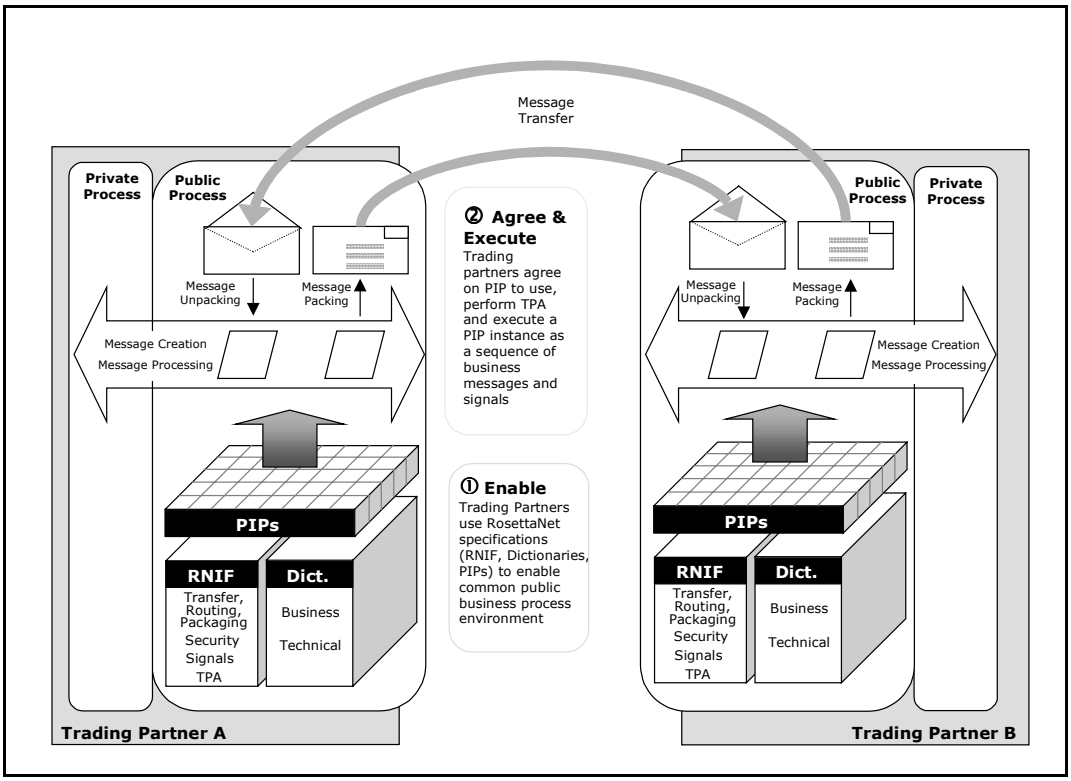


Figure 1. RosettaNet Specifications in a Trading Partner Implementation

1.1.2 Scalability of RosettaNet Specifications

As RosettaNet specifications are increasingly implemented within trading partners' enterprises, the issue of scalability (for increasing volumes) and applicability to related e-business transactions that are not directly addressed by current RosettaNet supply-chain-specific PIPs arises.

Similarly, solution partners face the challenge of creating and maintaining products that must support multiple approaches and sets of specifications to e-business within many supply chains.

Therefore, RosettaNet has recognized the need for increasing members' ability to interoperate across supply chains and achieve greater proliferation of e-business processes. The approach to achieving this is to search for, foster, and participate in those industry initiatives that are designed to support a wider set of businesses. This is particularly true in the implementation framework arena.

For this version of the RosettaNet Implementation Framework, which is designed to support members' current implementation needs, particular attention has been paid to using existing well-tested industry standards wherever possible. Where there is no such existing standard, due recognition of the directions being taken by emerging cross-industry initiatives has informed the decisions reflected in this document.

The intent has been to pave the way for RosettaNet ultimately to converge with or adopt a broader framework, and therefore for members to gain the benefit of a more broadly applicable implementation.

1.2 Technical Background

This section introduces several key technical concepts and assumptions that pertain to all the RosettaNet specifications and are necessary to make effective use of the specification part of this document. See also the "Glossary" in this document.

1.2.1 Public vs. Private Processes

An organization's business processes can be divided into two broad categories. The business processes that are internal to the organization are called "private processes," while the business processes that involve interactions with trading partners are known as "public processes."

The public processes are business processes through which partners conduct e-business. Within the context of RosettaNet, these are the partner interface processes that are visible between trading partners. Public processes implement the RosettaNet PIP specifications to exchange standard business documents over standard Internet transfer protocols, as specified by the RosettaNet Implementation Framework.

Within trading partner enterprises, private processes interface with public processes and with back-end business systems as needed to facilitate e-business exchanges between trading partner organizations.

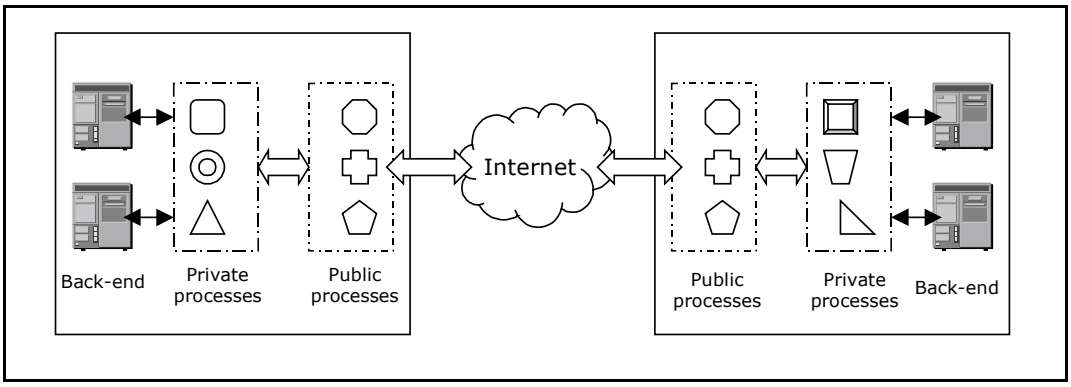


Figure 2. Private vs. Public Processes

1.2.1.1 Interoperability Considerations

For public processes to be interoperable, the information format and the sequence of message exchanges as executed by the public processes must conform to RosettaNet specifications. However, organizations may wish or need to implement new private processes or modify existing private processes (that mesh the back-end systems to the public processes) for this purpose.

1.2.2 PIPs and the Implementation Framework

A major part of RosettaNet's standardization effort is alignment of business processes between trading partners in a given supply chain (such as the IT Products and Electronic Component supply chains). RosettaNet specifies these as Partner Interface Process (PIP) specifications.

RosettaNet divides the entire e-business supply chain domain for which PIPs are specified into broad classifications called "clusters." Each cluster is further sub-divided into two or more "segments." Each segment comprises several PIPs. PIPs contain one or more Activities, and Activities in turn specify Actions. An example of this relationship follows:

- CLUSTER 3: Order Management
  - Segment A: Quote and Order Entry
    - PIP 3A4: Manage Purchase Order
      - Activity: Create Purchase Order
        - Action: Purchase Order Request
  - Segment B: Transportation and Distribution
  - Segment C: Returns and Finance
  - Segment D: Product Configuration

Each PIP in a segment represents a well-defined business process subset, and is named with the cluster, segment, and sequence number of the PIP in the segment. For example the Manage Purchase Order PIP is fourth in sequence in Segment A (Quote and Order Entry) of the Cluster 3 (Order Management). Hence the Manage Purchase Order PIP is identified by the name PIP3A4.

PIPs include specification of partner business roles (Buyer, Seller etc.); business activities involved between the roles; and type, content, and sequence of business documents exchanged by the role-interactions while performing these activities. They also specify the time, security, authentication, and performance constraints of these interactions. Structure and content of the business documents exchanged is specified through XML Document Type Definitions (DTDs) and associated Message Guidelines.

Trading partners that participate in the PIP exchange business documents that conform to the DTDs and Message Guidelines in the subject PIP specification, using network protocols that are specified and supported by the RosettaNet Implementation Framework.

Figure 3 is an example PIP interaction diagram that shows the business roles, messages, and their sequence of exchange in the PIP.

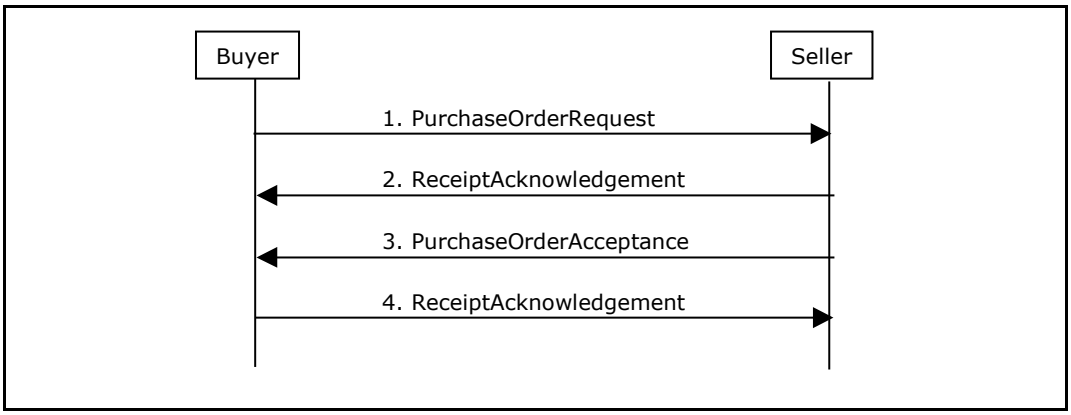


Figure 3. Sample PIP Interaction Diagram

1.2.2.1 Action and Signal Messages

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

Business actions are messages with content that is of a business nature, such as a Purchase Order or a Request For Quote. The DTDs and the associated Message Guidelines for business actions are specified as part of the corresponding PIP specification.

Business signals are positive and negative acknowledgment messages that are sent in response to business actions. Business signals are specified by and are part of the RosettaNet Implementation Framework. RNIF 2.0 contains one positive and one negative business signal.

Note: Only business actions are acknowledged. Business signals are never acknowledged.

POSITIVE SIGNALS

Receipt-Acknowledgment: This message is a positive acknowledgment of receipt of a Business Action message. Sent when an action message is received by the trading partner and is found to be a structurally and syntactically valid RosettaNet business action message. This message is sent only if it is required by the PIP and it is almost always required.

Note: In RNIF 2.0, RosettaNet eliminated the Acceptance Acknowledgment Signal, which had not been used in any of the PIPs.

The PIP specification that specifies the business actions also specifies which business signals are required. In section 2.6, RNIF provides detailed guidelines for PIP developers regarding when a specific kind of signal should be sent.

NEGATIVE SIGNALS

Exception: This business signal is a negative acknowledgment message that is sent to indicate an error. (See also the Notification of Failure PIP in section 2.5.)

In RNIF 2.0, there is only one exception message (versus three in RNIF 1.1). In RNIF 2.0, individual exceptions have been converted to exception types within the same exception signal. This change allows for faster implementation of additional or changed types. The following "exception types" are equivalent to the separate exceptions that were used in RNIF 1.1.

- *Receipt-Acknowledgment-Exception*: This is a negative acknowledgment of receipt of a business action message. It is sent when a message is received by the trading partner and is found to be a structurally or syntactically invalid RosettaNet business action message.
- *General-Exception*: This is a negative acknowledgment message that is sent to indicate an error other than the above. For example, in RNIF 1.1 this signal was sent when an error was detected during sequence validation or while performing the requested action. (See also the Notification of Failure PIP in section 2.5.)

RosettaNet recommends that authentication or authorization failures should not be responded to with exception messages. This is to minimize the risk of security attacks. See section 2.3.4 for further details.

1.2.3 PIP Message Exchange Models

Current PIP specifications are based on a Peer-to-Peer business message exchange model, between the RosettaNet networked applications (and hence the trading partners). That is, RosettaNet messages are exchanged between two trading partners directly. This peer-to-peer mode of message exchange relies on prior knowledge of the peer network entity identities and their addresses, which should be exchanged by the trading partners in advance. In RNIF 2.0, RosettaNet is introducing a mechanism to facilitate exchange of these messages through a third-party routing entity such as a hub (a.k.a. intermediary). However this mechanism is still based on the peer-to-peer message exchange model as far as the PIP is concerned. That is, the business entities involved in the exchange are still two: the originator and the final recipient, with the Hub simply facilitating the routing and delivery of the messages. RosettaNet is investigating other message exchange models for potential future use by PIP specifications. These include: Broadcast to all trading partners together; Publish and Subscribe mode of message exchanges between trading partners; and Multicast to a select subset of the trading partners.

1.2.4 PIP Metamodel

A PIP specification includes three major parts. These are the Business Operational View (BOV), the Functional Service View (FSV), and the Implementation Framework View (IFV).

Each PIP performs one or more discrete business activities, as specified in the PIP blueprints by the business community. These activities are identified in the BOV of

the PIP specification as described below. For example, the BOV of PIP 3A4 shows three separate business activities: Create Purchase Order, Change Purchase Order, and Cancel Purchase Order.

Each activity in the BOV translates into Business Actions and Signals that are exchanged between network components as specified in the FSV part of the PIP specification as described below. The IFV specifies the format (XML) and the corresponding guidelines for the actions and is further described below.

1.2.4.1 Business Operational View (BOV)

The Business Operational View (BOV) of a PIP specification captures the semantics of business entities and the flow of business information between Roles involved in the exchange as they perform business activities. The content of the BOV section of a PIP specification is based on the PIP Blueprint document created for RosettaNet's business community.

Figure 4 is an example BOV flow diagram (using PIP 2A3, "Query Marketing Information").

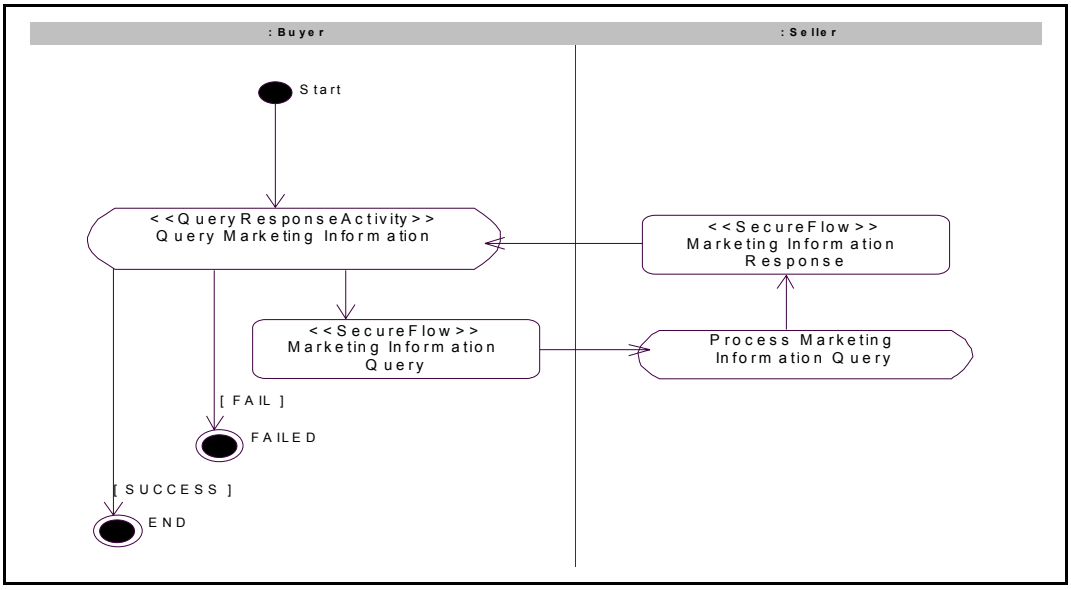


Figure 4. Sample BOV Flow (Using "Query Marketing Information" PIP)

The diagram shows that the PIP involves the exchange of business information between "Buyer" and "Seller" Roles. The specific activity involved in the PIP is "Query Marketing Information" and it is a "QueryResponseActivity" type of activity. The flow also shows that "Query Marketing Information" activity involves the flow of the "Marketing Information Query" business action from the "Buyer" to the "Seller" and a subsequent flow of the "Marketing Information Response" business action from the "Seller" to the "Buyer". The <<Secure Flow>> stereotype in the boxes containing the business actions implies that the business action MUST be transported from sender to recipient in a secure way.

The BOV part of the PIP specification also contains the description and type of the Business Roles involved in the BOV flow. A role type can be one of Organizational, Employee, or Functional. When two trading partners execute a business process within the RosettaNet framework, each partner performs a role. As the name implies the "Organizational" role is for playing the role of an "organization" such as an enterprise, a company, or a factory to cite few examples. The "Employee" role is used in business interactions that are performed by employees of an organization. The "Functional" role is for the cases when the interaction can be performed by either an employee or an organization.

The Business Activity Control section of the BOV contains business activity performance control specifications. For each activity in the PIP, this section specifies whether a "Receipt Acknowledgment" is required; if so, it also specifies whether it should be a non-repudiable acknowledgment and the time within which the acknowledgment should be sent. This section also contains other specifications, such as whether "Authorization is Required" to perform the activity.

Refer to the PIP specification for complete details of the BOV part of that PIP specification.

1.2.4.2 Functional Service View (FSV)

The Functional Service View (FSV) part of a PIP specification is derived from the BOV and specifies the network component design and the interactions between the network components as they execute the PIP. The network components specified in this section of the PIP are also known as the RosettaNet "services."

Note: In RNIF 2.0 onwards, the "agent" network component and related interaction dialogs have been removed from the Functional Service View part of the PIP specifications. See Appendix C of this document for details.

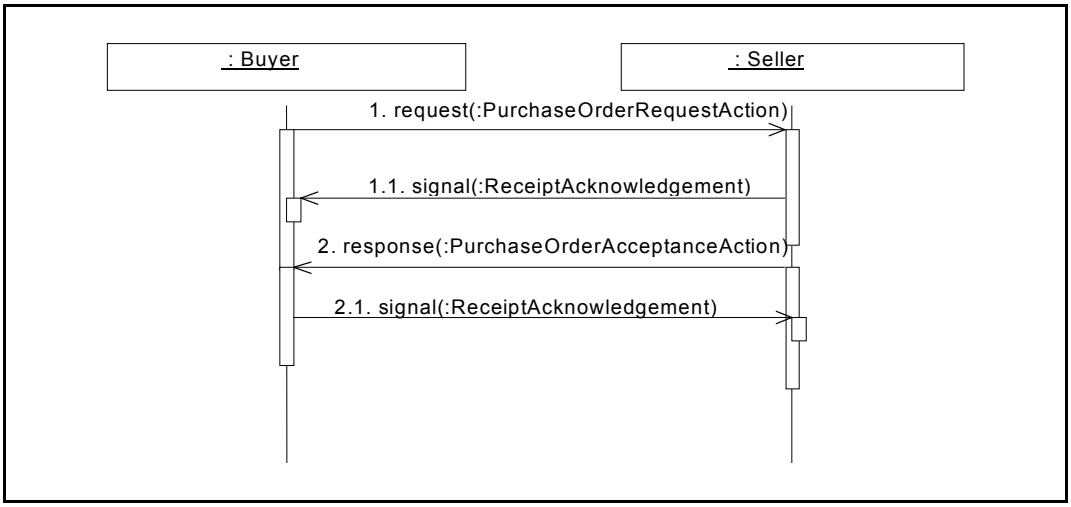


Figure 5. Sample FSV Network Component Dialog

Figure 5 identifies “Buyer” and “Seller” as two RosettaNet services (network components). It also depicts the interactions between them, namely, the “request” and “response” actions and the corresponding Receipt-Acknowledgment signals.

The FSV also defines the message exchange controls for each of the actions and signals involved in the dialog. For actions, this includes specification of time within which an Acknowledgment of Receipt signal should be sent; time within which a response to the action should be sent (if applicable); whether authorization is required to perform the action; and whether a secure transport should be used to transmit the action to the recipient.

Refer to the PIP specification for complete details of the FSV part of that PIP specification.

1.2.4.3 Implementation Framework View (IFV)

The Implementation Framework View (IFV) specifies the action message formats and communication requirements between network components as supported by the RosettaNet Implementation Framework. The communication requirements include specifications on the requirement for secure transport protocols such as SSL and digital signatures. For message formats, RosettaNet distributes XML DTDs and Message Guidelines for the action messages that are exchanged when the PIP is executed.

The RNIF 2.0-compliant PIP specifications include the BOV and FSV specifications and the XML Message Guidelines part of the IFV. However, other aspects of IFV such as the communications requirements between peer network components are no longer specified in the PIP specification, as these aspects can be derived from the BOV and FSV parts of the PIP specification in a well-defined and consistent fashion. Refer to ~~Appendix B~~Appendix C in this document for a description of how the BOV and FSV sections of a PIP specification can be mapped to such Implementation Framework View (IFV) aspects.

1.2.5 RosettaNet Business Message Overview

This section introduces the complete RosettaNet Business Message, as well as other parts of a completely packaged business message.

1.2.5.1 Parts of a RosettaNet Business Message

The individual business documents involved in a PIP (i.e., action and signal messages) are exchanged in a container that packs together other related entities such as headers, attachments 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.” Section 2 of this document gives the complete specification of the RosettaNet Business Message format and the corresponding packaging and unpackaging aspects. Below is an introduction to the basic structure and components of the RosettaNet Business Message.

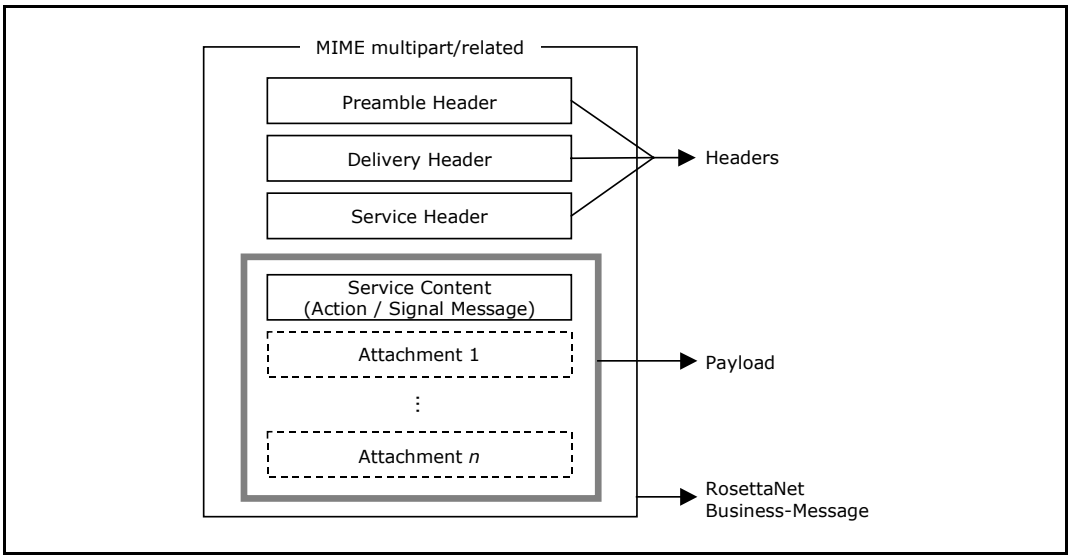


Figure 6. Parts of a RosettaNet Business Message

A RosettaNet Business Message always contains a Preamble header, a Delivery Header, a Service Header, and a Service Content. Service Content comprises an action message or a signal message. If Service Content is an action message, one or more attachments may be included. As shown, the headers and Service Content are packaged together using a MIME multipart/related construct. (This is similar to the RNIF 1.1 packaging scheme.) A RosettaNet Business Message can optionally be digitally signed. In RNIF 1.1, the RosettaNet Object (RNO) format was used for this purpose. However RNIF 2.0 does away with the RNO format and uses the standard S/MIME mechanism in its place. Refer to section 2 for details on the use of S/MIME for digital signatures and also for complete details of the Preamble and Service Header and their constituent elements.

1.2.5.2 Third-Party (Non-RosettaNet) Service Content

As described above, the Service Content contains either an action message or a signal message. A signal message must always be a RosettaNet-defined signal message instance. However, for action messages, RNIF 2.0 provides the option of shipping business action messages in a third-party defined format. The RNIF 2.0 Service Header now includes additional fields that facilitate this. For example, the header now includes fields that identify the "standard body" and the "version" of the specification to which the action message conforms.

Only action messages (also known as "business content") can be of non-RosettaNet origin. These messages must still be exchanged in a RosettaNet-defined PIP and must be sanctioned by RosettaNet by explicit identification of the sanctioned third-party action messages, in the PIP specification. Additionally, trading partners need to agree in advance to exchange third-party business content (for example, through a Trading Partner Agreement). This agreement would include the PIP payload binding information (i.e., which third-party business content would be used as a replacement for a particular action message in a PIP).

If this feature is not made available in a solution, the solution will not be deemed non-compliant. Similarly, a receiving trading partner MAY not wish to use this feature. This is also acceptable.

Refer to section 2 and to Appendix C for complete details.

1.2.5.3 Routing RosettaNet Business Messages through Hubs

In this version of RNIF, trading partners have the option of exchanging business messages directly with each other or through intermediary third-party routers (such as hubs).

To facilitate routing messages through hubs, RNIF 2.0 introduces a new type of header called the Delivery Header. The Delivery Header contains elements for the sending and receiving trading partner identities, the date and time stamp of the message, and a globally unique tracking ID. An instance of the Delivery Header is always present in a RosettaNet Business message and MUST be added by the initiator of the message.

All parties involved in routing the message from its originating point to the (eventual) destination, including any intermediaries if involved, use the information in the Delivery Header.

In RNIF 2.0, parts of the RosettaNet Business Message can be encrypted, including the Service Content and Service Header parts. In order for third-party hubs that may not have access to the encrypted Service Header to be able to route the message, the delivery-related elements are now part of the Delivery Header, which is never encrypted.

The tracking ID element of the Delivery Header and the message creation date and time stamp element help all parties involved in the message path to track the message in a globally unique fashion.

The Delivery Header also contains elements for specification of other requirements, such as whether a secure transport must be used to transmit the message between the nodes.

Note all headers namely, Preamble, Delivery and Service Headers, are always present in the message with only one instance of each (see Figure 6). Specifically, there is always one instance of the Delivery Header, as it is created by the originator/sender of the message and stays unaltered (along with all other components of the message) as it is routed and delivered to the final recipient.

For more details on the Delivery Header please refer to section 2 of this specification.

1.2.6 Signals vs. Process Control PIPs

Signals are used between two peers to communicate certain “events” within a PIP instance, such as “receipt and successful validation of a message” (Receipt Acknowledgment), “receipt of an out of sequence message” (Exception with a type of

"General Exception"), or "receipt of a message that has invalid grammar" (Exception with a type of "Receipt Acknowledgment Exception").

Process Control PIPs, on the other hand, are used to communicate process states outside of the context of the current process instance. An example is the 0A1 "Notification of Failure" PIP. A new instance of the 0A1 process is started when exceptions happen under a specific condition (namely, when the process is in "execution" state at one partner's system and may have possibly reached a "completed" state in the other partner's system) during the execution of any other process.

1.2.7 Network Application Model

The RNIF specifies the transfer and security level protocols to be used and the format of the RosettaNet business messages that are exchanged by the networked applications. The following diagram captures the RosettaNet networked application protocol stack when exchanging RosettaNet business messages.

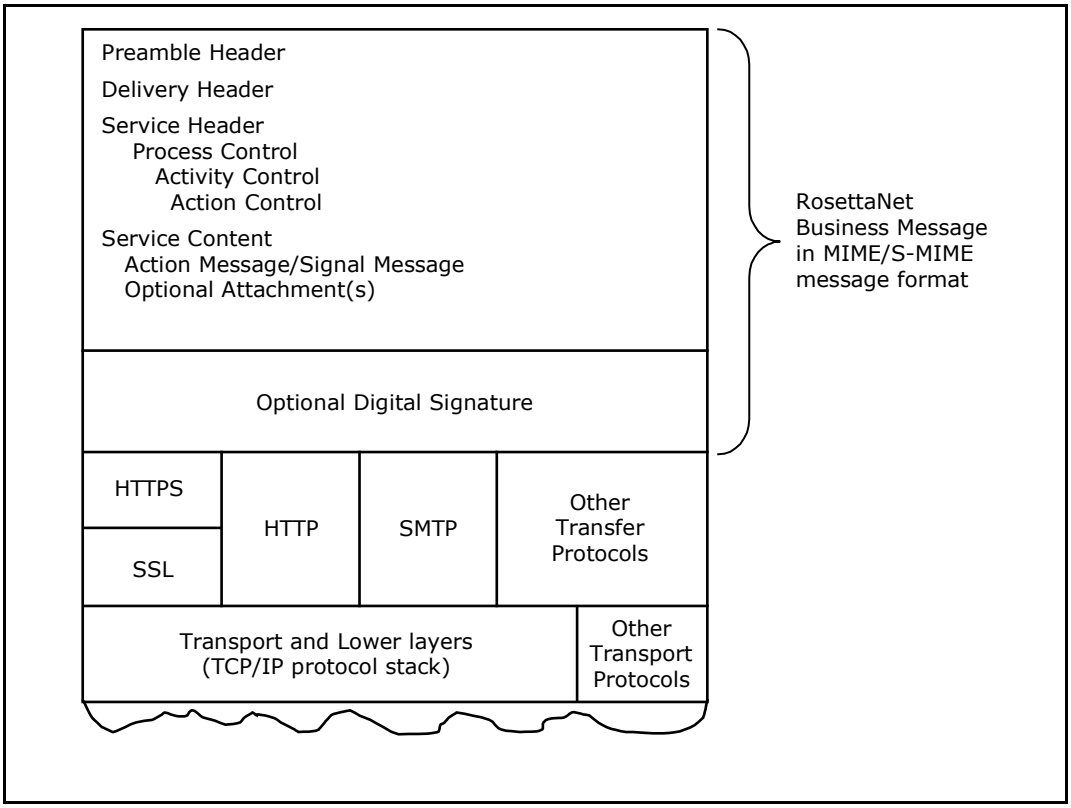


Figure 7. Network Application Model

1.2.8 Authentication, Authorization and Non-Repudiation

This section explains the concepts of "authentication," "authorization," and "non-repudiation" within the context of RNIF 2.0.

1.2.8.1 Authentication

Authentication within the context of RNIF 2.0 is the act of making sure that the sender of a RosettaNet Business Message is who the sender claims to be. This is accomplished by requiring the sender of the message to digitally sign the message. In RNIF 2.0, a RosettaNet Business Message is digitally signed following the S/MIME IETF (RFC 2311) specification. See section 2.2 for further details.

The PIP specifications specify whether the messages exchanged must be digitally signed. If so, then the sending partner is required to digitally sign the messages sent to its partner. The receiving partner authenticates the message sender by following the standard S/MIME and PKCS mechanisms to verify the digital signatures. See section 2.2 for more details.

1.2.8.2 Authorization

Authorization is the act of making sure that the sender of a message is permitted or authorized to send the subject message to the receiving partner. The requirement on Authorization of message exchanges in PIP is specified in the corresponding PIP specification. The trading partners must establish agreement between themselves in advance, by identifying the PIPs they would execute between themselves and the Digital Certificates that would be used to sign the messages exchanged. Each message exchanged must also be digitally signed using the S/MIME mechanism as described earlier.

Authorization is typically a two-step process. The first step is making sure that the sending partner (as identified in the Delivery and Service Headers) is authorized to send the subject message (PIP). The second step is making sure that the sending partner's organization, as identified by the digital signature on the message, is authorized to send the subject message.

See section 2.2 for further details.

1.2.8.3 Non-Repudiation

Non-Repudiation is the mechanism for making sure that an originating trading partner can not deny having originated and sent a message (called "Non-Repudiation of Origin and Content") and that a receiving trading partner cannot deny having received a message sent by its partner (called "Non-Repudiation of Receipt"). Non-repudiation requirements are explicitly called out in PIP specifications.

NON-REPUDIATION OF ORIGIN AND CONTENT

For the purpose of Non-Repudiation of Origin and Content, the originating partner of a RosettaNet Business Message must digitally sign the message following the S/MIME mechanism as described earlier.

The partner receiving the RosettaNet Business Message must store the message in original form for a mutually agreed period of time (typically three to seven years).

This prevents an initiating partner from later denying that they originated contents of a Business Document.

NON-REPUDIATION OF RECEIPT

For the purpose of Non-Repudiation of Receipt, a signed Receipt-Acknowledgment signal must be sent for the received RosettaNet Business Message. The Acknowledgment message must be digitally signed and must also include an MD5 or SHA-1 digest of the message being acknowledged. Additionally the partner receiving the acknowledgment must store the receipt and original message in their original form for a mutually agreed period of time (typically three to seven years). This prevents a responding partner from later denying that they received a Business Document.

2 Technical Specifications

This section contains the actual specifications approved by RosettaNet for constructing and exchanging RosettaNet Business Messages. It begins with the specifications for the various components of a RosettaNet Business Message, proceeds to the packaging (and unpackaging) of such messages, and then specifies the various transfer mechanisms for exchanging those messages. It also contains specifications for security, for process control PIPs, and for RosettaNet business signals. Additionally, a section on message flow is included.

2.1 RosettaNet Business Message Components

This section enables the implementer to understand what is needed to populate the various parts of the RosettaNet Business Message. For simplicity, this section also includes specifications for special headers needed to route the RosettaNet Business Message for trading partners using an intermediary service provider (e.g., a hub).

2.1.1 Introduction

A RosettaNet Business Message consists of various components as shown in Figure 6. Excepting attachments (if any), all the components in the RosettaNet Business Message are XML documents.

This section provides the syntax, semantics, and descriptions for the various business message components, such as the various headers used to transmit a RosettaNet action message or a RosettaNet business signal. Compliant implementations MUST adhere to these syntactic and semantic rules in order to ensure interoperability.

This section only describes the XML headers for action or signal messages. It does not include the MIME headers used for packaging or the transfer headers used with a particular transfer protocol. Refer to those appropriate sections (2.3 and 2.4) for information regarding the MIME headers and the transfer headers.

2.1.2 XML Usage

Since the core of the RosettaNet Business Message is in XML, it is important to clarify the usage of XML in the context of encoding and element validation.

2.1.2.1 Encoding Rules

For XML documents, RosettaNet permits both UTF-8 and UTF-16 encoding schemes. Senders MAY choose either encoding based on the content of the XML document. The receivers MUST be able to handle both encoding schemes. Subject to the constraints of the chosen transfer protocol, the XML parts MAY be MIME content-transfer encoded. See RFC 2376 and W3C's XML specification for details.

2.1.2.2 Validation Rules

All elements MUST be validated against the DTD for the document type that contains it, based on standard DTD grammar validation rules.

The following is the minimum level of validation that is required on each of the XML body parts, namely, the Preamble, the Delivery Header, the Service Header, and the Service Content.

1. The XML document MUST be compliant with its corresponding DTD.
2. Where an element's data type and/or length is specified in the corresponding RosettaNet Message Guideline, the element MUST be validated against these specifications.
3. Where an element's allowed list of values is specified in the Entity Instance list in the corresponding RosettaNet Message Guideline, the element MUST be validated against these specifications.
4. Where the cardinality specification of an element in the Message Guideline is different from the corresponding specification in the DTD, the specification in the Message Guideline is more accurate and MUST be adhered to.
5. Where the sequence or naming of an element in the Message Guideline is different from the corresponding specification in the DTD, the specification in the DTD is more accurate and MUST be adhered to.
6. Where a dictionary is present and the PIP requires Dictionary Validation, the Service Content MUST be validated against the dictionary as a part of action performance.
7. If a message does not follow one or more of the above rules, then it MUST be deemed invalid.

For elements with validation rules specified in the form of a list of valid or allowed values, all these values are case sensitive (where not specified otherwise). Also, these elements are to be treated as "white space sensitive."

For example, if the allowed values are "Action" and "Signal" for an element or attribute, then "action", " signal", "SIGNAL", and "A c t I o n" are all examples of incorrect usage. The only allowed values are those that match an entry in the code list exactly for case, spacing, and punctuation.

As a further example, suppose there is an element called "ShipToCountry". If the element is specified with a cardinality of 1, and if the only allowed value is "United States of America" then the following is the only allowed XML instance of this element.

```
<ShipToCountry>United States of America</ShipToCountry>
```

The following are examples of incorrect usage:

```
<ShipToCountry>United States Of America</ShipToCountry>
<ShipToCountry> United States Of America </ShipToCountry>
<ShipToCountry>UnitedStatesOfAmerica</ShipToCountry>
```

2.1.2.3 Constraints on Message Elements

The following constraints on RosettaNet-defined message elements have been identified:

- Instance identifiers

  Constraint: length constrained to maximum of 255 characters

- Date/time elements

  Elements that refer to date and time MUST follow the format for date and time as specified in the ISO 8601 specification. Specifically, RosettaNet has chosen the format: CCYYMMDDThhmmss.sssZ , where "CC" represents the century, "YY" the year, "MM" the month, and "DD" the day. The letter "T" is the date/time separator and "hh", "mm", and "ss.sss" represent hour, minute, and second respectively. The "Z" at the end of the date/time element indicates Coordinated Universal Time. All elements of this format MUST be present.

- Case sensitivity

  All element names and element values are case-sensitive.

2.1.2.4 DTD Naming, Pathname Specification and Versioning

All XML documents which are based on specifications that include an associated Document Type Definition (DTD) MUST reference the ~~associated~~ DTD by specifying the doctype element. The name of the DTD file as published by RosettaNet MUST be specified, and MUST NOT be renamed differently. The doctype element MUST NOT specify any additional URL qualifiers that refer to a specific location where the DTD file exists. Recipients of RosettaNet XML messages are responsible for configuring their systems to find the appropriate DTD file.

~~All DTD filenames MUST include the version number of the DTD in the format specified by the *RosettaNet Technical Conventions and Style Guide*.~~

~~All DTD filenames must include the version number of the DTD.~~

Example: 2A5_MS_R01_00_TechInfoQuery.dtd

2.1.2.5 XML Namespace

A namespace attribute is present in all header~~s~~ and business signal DTD files: Preamble, Delivery Header, ~~and~~ Service Header, Exception and Receipt Acknowledgment.

This is a default attribute with the value "http://www.rosettanet.org/RNIF/V02.00~~/~~".

2.1.3 Header Structure and Format Specifications

This section describes the various headers that are sent along with a RosettaNet business action message or a RosettaNet business signal message. Each of these headers is an XML document, and each of them has a DTD.

The following are the various message headers:

- Preamble – This header identifies the standard with which this message structure is compliant.
- Delivery Header – This header identifies message sender and recipient and message instance information. This information is placed separately from the Service Header to allow access to the information by a Hub when the Service Header is encrypted.
- Service Header – This header identifies the PIP, the PIP instance, the activity, and the action to which this message belongs.

The overall purpose of these headers is for the recipient to be able to:

- Identify the message as a RosettaNet Business Message;
- Identify the context of the message;
- Identify the sender for authentication and authorization.

2.1.3.1 Preamble Specification

The Preamble is used to identify the standard and the version of the standard with which the message structure is compliant. All RosettaNet messages MUST have a Preamble. The structure of the Preamble MUST follow the Preamble DTD.

The values of the elements in the Preamble are fixed by the sender of the first message in the Activity. All subsequent messages in the activity MUST NOT change the contents of the preamble.

DOCUMENT TYPE DEFINITION

```
<!ENTITY % common-attributes "id  CDATA #IMPLIED" >
<!ELEMENT Preamble (
          standardName ,
          standardVersion ) >
<!ATTLIST Preamble xmlns CDATA #FIXED
          "http://www.rosettanet.org/RNIF/V02.00" >
<!ELEMENT standardName
           ( GlobalAdministeringAuthorityCode ) >
<!ELEMENT GlobalAdministeringAuthorityCode
           ( #PCDATA ) >
<!ELEMENT standardVersion
           ( VersionIdentifier ) >
<!ELEMENT VersionIdentifier
           ( #PCDATA ) >
```

TREE STRUCTURE FROM MESSAGE GUIDELINE

```
1  1  Preamble
2  1  |-- standardName.GlobalAdministeringAuthorityCode
3  1  |-- standardVersion.VersionIdentifier
```

ELEMENT DESCRIPTION

Table 1 provides descriptions of the Preamble elements and special validation and processing rules where applicable. Note that the Element Names have one-to-one correspondence with the Element Tag Names, but are not exactly the same. The element names have been formatted for readability, and white spaces have been introduced. The official element descriptions appear in the separately published Message Guideline associated with the Preamble DTD.

Table 1. Preamble Elements

Note: This table is provided to assist in understanding how this header works. For complete documentation on these elements, consult the Message Guideline itself.

Element	Description/Notes	Special Validation and Processing Rules
Global Administering Authority Code	Instance from set of codes identifying administrating authority.	
Standard Name	Identifies the name of the standard with which this message structure is compliant.	In the case of a RosettaNet-compliant message, the only allowed value is "RosettaNet".
Standard Version	Identifies the version number of the standard.	When the Standard Name is "RosettaNet", the Standard Version MUST carry the version number of the RNIF specification. For a message compliant with RNIF 02.00, this value MUST be "V02.00".

Example 1. Preamble Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Preamble SYSTEM "Preamble_MS_~~B~~V02_00.dtd">
<Preamble>
   <standardName>
      <GlobalAdministeringAuthorityCode>
      RosettaNet
      </GlobalAdministeringAuthorityCode>
   </standardName>
   <standardVersion>
      <VersionIdentifier>V02.00</VersionIdentifier>
   </standardVersion>
</Preamble>
```

VERSIONING NOTES

RNIF 2.0 invalidates the 1.1 version of the Preamble. The new version to use is version 2.0 of the Preamble, which follows the DTD structure cited in this section.

COMPLIANCE SUMMARY

This summary is for convenience only and is not guaranteed to contain all compliance statements. For complete compliance knowledge, read the entire specification.

A message that is compliant with RNIF 2.0 MUST have an XML document called Preamble. This document MUST have been packaged according to the packaging rules specified in section 2.3. The document MUST conform to the DTD cited above and MUST have values in conformance to the applicable Message Guideline.

2.1.3.2 Delivery Header Specification

This header is added as a separate MIME part to specify route and message instance information. This information is placed separately from the Service Header to allow access to the information by a Hub when the Service Header is encrypted.

DOCUMENT TYPE DEFINITION

```
<!ENTITY % common-attributes "id  CDATA #IMPLIED" >
<!ELEMENT DeliveryHeader (
          isSecureTransportRequired ,
          messageDateTime ,
          messageReceiverIdentification ,
          messageSenderIdentification ,
          messageTrackingID ) >
<!ATTLIST DeliveryHeader xmlns CDATA #FIXED
          "http://www.rosettanet.org/RNIF/V02.00" >
<!ELEMENT isSecureTransportRequired ( AffirmationIndicator ) >
<!ELEMENT AffirmationIndicator ( #PCDATA ) >
<!ELEMENT messageDateTime ( DateTimeStamp ) >
<!ELEMENT DateTimeStamp ( #PCDATA ) >
<!ELEMENT messageReceiverIdentification ( PartnerIdentification ) >
<!ELEMENT PartnerIdentification
          ( domain? ,
          GlobalBusinessIdentifier ,
          locationID? ) >
<!ELEMENT domain ( FreeFormText ) >
<!ELEMENT FreeFormText ( #PCDATA ) >
<!ATTLIST FreeFormText xml:lang CDATA #IMPLIED >
<!ELEMENT GlobalBusinessIdentifier ( #PCDATA ) >
<!ELEMENT locationID ( ~~FreeFormText~~ Value ) >
<!ELEMENT messageSenderIdentification ( PartnerIdentification ) >
<!ELEMENT messageTrackingID ( InstanceIdentifier ) >
<!ELEMENT InstanceIdentifier ( #PCDATA ) >
<!ELEMENT Value ( #PCDATA ) >
```

TREE STRUCTURE FROM MESSAGE GUIDELINE

```
1  1     DeliveryHeader
2  1     |-- isSecureTransportRequired.AffirmationIndicator
3  1     |-- messageDateTime.DateTimeStamp
4  1     |-- messageReceiverIdentification.PartnerIdentification
5  0..1  | |-- domain.FreeFormText
6  1     | |-- GlobalBusinessIdentifier
7  0..1  | |-- locationID.~~FreeFormText~~ Value
8  1     |-- messageSenderIdentification.PartnerIdentification
9  0..1  | |-- domain.FreeFormText
```

```
10 1     | |-- GlobalBusinessIdentifier
11 0..1  | |-- locationID.~~FreeFormText~~ Value
12 1     |-- messageTrackingID.InstanceIdentifier
```

ELEMENT DESCRIPTION

Table 2 provides descriptions of the Delivery Header elements and special validation and processing rules where applicable. Note that the Element Names have one-to-one correspondence with the Element Tag Names, but are not exactly the same. The element names have been formatted for readability and white spaces have been introduced. (For example, the Element Name "Sent To" in the table corresponds to the element with the tag name "SentTo".) The official element descriptions appear in the separately published Message Guideline associated with the Delivery Header DTD.

Table 2. Delivery Header Elements

Note: This table is provided to assist in understanding how this header works. For complete documentation on these elements, consult the Message Guideline itself.

Element Name	Description/Notes	Special Validation and Processing Rules
Affirmation Indicator	Used to indicate "Y~~y~~es" or "N~~n~~o" statements ~~(e.g., Serialized Product)~~.	Valid values are "Yes" or "No".
Date Time Stamp	Specifies an instance in time.	
Domain	Identifies the area of applicability. (In this case, identifies content of the Partner ID, e.g., whether or not is DUNS.	For RNIF 2.0, the only allowed value is "DUNS". If this optional element is not present, the default is "DUNS".
Free Form Text	Unformatted text.	
Global Business Identifier	The DUNS number of the trading partner.	
Instance Identifier	A unique alphanumeric identifier that represents a specific instance of a business process, business transaction, business action, or business signal. The instance identifier must be unique for a particular instance of a business process, business transaction, business action and business signal.	
Is Secure Transport Required	Affirmative value indicates that the next hub must transmit this message securely.	~~Valid values are "Yes" or "No".~~
Location ID	Identifies a logical business location associated with the trading partner.	
Message Date Time	The date and time associated with the creation of a message.	The timestamp MUST be generated as close to the time of first attempted transport as possible.

Element Name	Description/Notes	Special Validation and Processing Rules
Message Tracking ID	Uniquely identifies the message for tracking purposes.	MUST be unique within the context of the message sender.
Message Receiver Identification	~~Global Business Identifier~~ Identity of party receiving message~~, and an optional "location ID"~~.	
Message Sender Identification	~~Global Business Identifier~~ Identity of party sending message~~, and an optional "location ID"~~.	
Partner Identification	Identifies a trading partner associated with this message by Global Business Identifier and optional Location ID.	
Value	Identifies the locationID.	

Example 2. Delivery Header Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DeliveryHeader SYSTEM "DeliveryHeader_MS_BV02_00.dtd">
<DeliveryHeader>
<isSecureTransportRequired>
   <AffirmationIndicator>yYes</AffirmationIndicator>
</isSecureTransportRequired>
<messageDateTime>
   <DateTimeStamp>20001121T145200.000Z</DateTimeStamp>
</messageDateTime>
<messageReceiverIdentification>
   <PartnerIdentification>
      <domain>
         <FreeFormText>DUNS</FreeFormText>
      </domain>
      <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      <locationID>
         <FreeFormTextValue>Santa Clara</FreeFormTextValue>
      </locationID>
   </PartnerIdentification>
</messageReceiverIdentification>
<messageSenderIdentification>
   <PartnerIdentification>
      <GlobalBusinessIdentifier>555123456</GlobalBusinessIdentifier>
      <locationID>
         <FreeFormTextValue>Hong Kong</FreeFormTextValue>
      </locationID>
   </PartnerIdentification>
</messageSenderIdentification>
<messageTrackingID>
   <InstanceIdentifier>543543</InstanceIdentifier>
</messageTrackingID>
</DeliveryHeader>
```

VERSIONING NOTES

This header is new in RNIF 2.0.

COMPLIANCE SUMMARY

This summary is for convenience only and is not guaranteed to contain all compliance statements. For complete compliance knowledge, read the entire specification.

An instance of this header MUST be added to the message being routed by the initiating node.

The received message MUST NOT be modified in any form by the intermediary nodes.

2.1.3.3 Service Header

The Service Header provides the process context for a message. It also provides information about ~~the sender of the message, the recipient of the message,~~ whether the message is a Test message or a Production message, who the PIP initiator is, ~~and~~ whether the ~~sender~~ initiator is ~~to be treated as~~ a~~n~~ known or "~~unknown~~ partner~~."~~, and Quality of Service negotiation information (which is currently unused).

DOCUMENT TYPE DEFINITION

```
<!ENTITY % common-attributes "id  CDATA #IMPLIED" >
<!ELEMENT ServiceHeader ( ProcessControl ) >
<!ATTLIST ServiceHeader xmlns CDATA #FIXED
          "http://www.rosettanet.org/RNIF/V02.00~~/~~" >
<!ELEMENT ProcessControl (
          ActivityControl ,
          GlobalUsageCode ,
          partnerDefinedPIPPayloadBindingId? ,
          pipCode ,
          pipInstanceId ,
          pipVersion ,
          QualityOfServiceSpecification?,
          ( KnownInitiatingPartner |
           UnknownInitiatingPartner ) ) >
<!ELEMENT ActivityControl (
          BusinessActivityIdentifier ,
          MessageControl ) >
<!ELEMENT BusinessActivityIdentifier ( #PCDATA ) >
<!ELEMENT MessageControl (
          fromRole ,
          fromService ,
          inReplyTo? ,
          Manifest ,
          toRole ,
          toService ) >
<!ELEMENT fromRole ( GlobalPartnerRoleClassificationCode ) >
<!ELEMENT GlobalPartnerRoleClassificationCode ( #PCDATA ) >
<!ELEMENT fromService ( GlobalBusinessServiceCode ) >
<!ELEMENT GlobalBusinessServiceCode ( #PCDATA ) >
<!ELEMENT inReplyTo ( ActionControl ) >
<!ELEMENT ActionControl (
          ActionIdentity ,
          messageTrackingID ) >
<!ELEMENT ActionIdentity (
          GlobalBusinessActionCode ,
          messageStandard? ,
```



```
          standardVersion?,
          VersionIdentifier ) >
<!ELEMENT GlobalBusinessActionCode ( #PCDATA ) >
<!ELEMENT messageStandard ( FreeFormText ) >
<!ELEMENT FreeFormText ( #PCDATA ) >
<!ATTLIST FreeFormText xml:lang CDATA #IMPLIED >
<!ELEMENT standardVersion ( VersionIdentifier ) >
<!ELEMENT VersionIdentifier ( #PCDATA ) >
<!ELEMENT messageTrackingID ( InstanceIdentifier ) >
<!ELEMENT InstanceIdentifier ( #PCDATA ) >
<!ELEMENT Manifest (
          Attachment* ,
          numberOfAttachments ,
          ServiceContentControl ) >
<!ELEMENT Attachment (
          description? ,
          GlobalMimeTypeQualifierCode ,
          UniversalResourceIdentifier ) >
<!ELEMENT description ( FreeFormText ) >
<!ELEMENT GlobalMimeTypeQualifierCode ( #PCDATA ) >
<!ELEMENT UniversalResourceIdentifier ( #PCDATA ) >
<!ELEMENT numberOfAttachments ( CountableAmount ) >
<!ELEMENT CountableAmount ( #PCDATA ) >
<!ELEMENT ServiceContentControl (
          ( ActionIdentity |
           SignalIdentity ) ) >
<!ELEMENT SignalIdentity (
          GlobalBusinessSignalCode ,
          VersionIdentifier ) >
<!ELEMENT GlobalBusinessSignalCode ( #PCDATA ) >
<!ELEMENT toRole ( GlobalPartnerRoleClassificationCode ) >
<!ELEMENT toService ( GlobalBusinessServiceCode ) >
<!ELEMENT GlobalUsageCode ( #PCDATA ) >
<!ELEMENT KnownInitiatingPartner ( PartnerIdentification ) >
<!ELEMENT PartnerIdentification (
          domain? ,
          GlobalBusinessIdentifier ,
          locationID? ) >
<!ELEMENT domain ( FreeFormText ) >
<!ELEMENT GlobalBusinessIdentifier ( #PCDATA ) >
<!ELEMENT locationID ( FreeFormText Value ) >
<!ELEMENT UnknownInitiatingPartner (
          PartnerIdentification ,
          UniformResourceLocator ) >
<!ELEMENT UniformResourceLocator ( #PCDATA ) >
<!ELEMENT partnerDefinedPIPPayloadBindingId
           ( ProprietaryReferenceIdentifier ) >
<!ELEMENT ProprietaryReferenceIdentifier ( #PCDATA ) >
<!ELEMENT pipCode ( GlobalProcessIndicatorCode ) >
<!ELEMENT GlobalProcessIndicatorCode ( #PCDATA ) >
<!ELEMENT pipInstanceId ( InstanceIdentifier ) >
<!ELEMENT pipVersion ( VersionIdentifier ) >
<!ELEMENT QualityOfServiceSpecification ( QualityOfServiceElement+ ) >
<!ELEMENT QualityOfServiceElement (
          QualityOfServiceClassificationCode ,
          Value ) >
<!ELEMENT QualityOfServiceClassificationCode ( #PCDATA ) >
<!ELEMENT Value ( #PCDATA ) >
```

TREE STRUCTURE FROM MESSAGE GUIDELINE

```
1  1     ServiceHeader
2  1     |-- ProcessControl
3  1     | |-- ActivityControl
4  1     | | |-- BusinessActivityIdentifier
5  1     | | |-- MessageControl
6  1     | | | |-- fromRole.GlobalPartnerRoleClassificationCode
7  1     | | | |-- fromService.GlobalBusinessServiceCode
8  0..1  | | | |-- inReplyTo.ActionControl
9  1     | | | | |-- ActionIdentity
10 1     | | | | | |-- GlobalBusinessActionCode
11 0..1  | | | | | |-- messageStandard.FreeFormText
12 0..1  | | | | | |-- standardVersion.VersionIdentifier
~~13 1     | | | | | |-- VersionIdentifier~~
134 1     | | | | |-- messageTrackingID.InstanceIdentifier
145  1     | | | |-- Manifest
156  0..n  | | | | |-- Attachment
167  0..1  | | | | | |-- description.FreeFormText
178  1     | | | | | |-- GlobalMimeTypeQualifierCode
189  1     | | | | | |-- UniversalResourceIdentifier
2019 1     | | | | |-- numberOfAttachments.CountableAmount
201  1     | | | | |-- ServiceContentControl
212  1     | | | | | |-- Choice
223        | | | | | | |-- ActionIdentity
234 1     | | | | | | | |-- GlobalBusinessActionCode
245 0..1  | | | | | | | |-- messageStandard.FreeFormText
256 0..1  | | | | | | | |-- standardVersion.VersionIdentifier
~~27 1     | | | | | | | |-- VersionIdentifier~~
268        | | | | | | | |-- SignalIdentity
279  1     | | | | | | | | |-- GlobalBusinessSignalCode
2830 1     | | | | | | | | |-- VersionIdentifier
2931 1     | | | | |-- toRole.GlobalPartnerRoleClassificationCode
302  1     | | | |-- toService.GlobalBusinessServiceCode
313  1     | | |-- GlobalUsageCode
324  0..1  | | |-- partnerDefinedPIPPayloadBindingId.Proprietary
                ReferenceIdentifier
335  1     | | |-- pipCode.GlobalProcessIndicatorCode
346  1     | | |-- pipInstanceId.InstanceIdentifier
357  1     | | |-- pipVersion.VersionIdentifier
368  0..1  | | |-- QualityOfServiceSpecification
379  1..n  | | | |-- QualityOfServiceElement
3840 1     | | | | |-- QualityOfServiceClassificationCode
3941 1     | | | | |-- Value
40234 1    | | |-- Choice
41335      | | | |-- KnownInitiatingPartner
42436 1    | | | | |-- PartnerIdentification
43537 0..1 | | | | | |-- domain.FreeFormText
44638 1    | | | | | |-- GlobalBusinessIdentifier
45739 0..1 | | | | | |-- locationID.~~FreeFormText~~Value
46840      | | | |-- UnknownInitiatingPartner
4791 1     | | | | |-- PartnerIdentification
485042  0..1  | | | | | |-- domain.FreeFormText
495143  1     | | | | | |-- GlobalBusinessIdentifier
50244 0..1 | | | | | |-- locationID.~~FreeFormText~~Value
51345 1    | | | | |-- UniformResourceLocator
~~46 0..1  | | |-- partnerDefinedPIPPayloadBindingId.Proprietary
                ReferenceIdentifier~~
~~47 1     | | |-- pipCode.GlobalProcessCode~~
~~48 1     | | |-- pipInstanceId.InstanceIdentifier~~
~~49 1     | | |-- pipVersion.VersionIdentifier~~
```

~~50 0..1 | |-- QualityOfServiceSpecification~~
~~51 1..n | | |-- QualityOfServiceElement~~
~~52 1 | | | | QualityOfServiceClassificationCode~~
~~53 1 | | | | Value~~

ELEMENT DESCRIPTION

Table 3 provides descriptions of the Service Header elements and special validation and processing rules where applicable. Note that the Element Names have one-to-one correspondence with the Element Tag Names, but are not exactly the same. The element names have been formatted for readability and white spaces have been introduced. (For example, the Element Name "PIP Code" in the table corresponds to the element with the tag name "PIPCode".) The official element descriptions appear in the separately published Message Guideline associated with the Service Header DTD.

Table 3. Service Header Elements

Note: This table is provided to assist in understanding how this header works. For complete documentation on these elements, consult the Message Guideline itself.

Element Name	Description / Notes	Special Validation and Processing Rules
Action Identity (In reply to)	The identity of the action to which this message is in reply.	
Action Control	Business action message control properties.	
Activity Control	Specifies the properties of this activity.	
Attachment	Details of the attachment. Not present if the number of attachments is zero. The number of entries for this element MUST be equal to the value specified in No Of Attachments.	
Business Activity Identifier	RosettaNet Activity identifier of this message.	
Countable Amount	Dimensionless magnitude, e.g., number of products.	
Description	A description of the attachment.	
Free Form Text	Unformatted text.	
From Role	The role that the trading partner sending this message plays in this PIP.	
From Service	The service from which this message is being sent.	

Element Name	Description / Notes	Special Validation and Processing Rules
Global Business Action Code (Action Identity)	The Action Code corresponding to the action to which this message is in reply.	For the valid value for this element, refer to the corresponding "InReplyToActionCode" element in the PIP Specification corresponding to the currently executing PIP, Activity, and Action.
Global Business Action Code (Service Content)	The Action Code if this is an action.	
Global Business Identifier	A unique business identifier. Use of the DUNS number is required by RosettaNet.	
Global Business Service Code (From Service and To Service)	The service specified in the PIP.	For the valid value for this element, refer to the corresponding "FromService" element (or "ToService" element, as the case may be) in the PIP Specification corresponding to the currently executing PIP, Activity, and Action. If the current message is a signal, then the value corresponding to the From Service in the signal MUST be the same as the value of the To Service in the action to which this signal is replying.
Global Business Signal Code (~~Service Content~~Signal Identity)	The Signal Code if this is a signal.	
Global Mime Type Qualifier Code	The MIME content type of the attachment.	This value MUST be picked from the MIME content type for the attachment.
Global Partner Role Classification Code (From Role and To Role)	The role specified in the PIP.	For the valid value for this element, refer to the corresponding "FromRole" element (or "ToRole" element, as the case may be) in the PIP Specification corresponding to the currently executing PIP, Activity, and Action. If the current message is a signal, then the value corresponding to the From Role in the signal MUST be the same as the value of the To Role in the action to which this signal is replying.
Global Process Indicator Code	Business process ~~identifier e.g. Manage Product Subscriptions~~. This code MUST be the PIP identifier (e.g., 3A4).	

Element Name	Description / Notes	Special Validation and Processing Rules
Global Usage Code	Determines whether this message is to be used in Test mode or in Production mode.	The only allowed values are "Test" and "Production".
In Reply To	The elements that help identify the message to which this message is in reply.	MUST be present if this is not the first message in an activity. MUST be present for all signals.
Instance Identifier	A unique alphanumeric identifier that represents a specific instance of an business process, business transaction, business action, or business signal. The instance identifier must be unique for a particular instance of a business process, business transaction, business action and business signal.	
Known Initiating Partner	A known partner initiating this PIP instance, with whom the responder has a valid TPA.	
Manifest	Provides a list of items in the payload section (i.e., the Service Content and the list of attachments if any).	
Message Control	The elements whose values change with every message in the PIP.	Note that all elements other than those in this group are set by the initiator and MUST remain the same through all messages in that PIP instance.
Message Tracking ID	Identifies the instance ID of the action to which this message is in reply.	Value MUST come from Message Tracking ID in the Delivery Header of the original received message.
Message Standard	The standard with which the Service Content MUST be compliant.	MUST be set if and only if this is a non-RosettaNet-specified Service Content message.
Number Of Attachments	The number of attachments.	If no attachments, the only allowed value is "0" (i.e., the number zero).
Partner-Defined PIP Payload Binding ID	MUST be specified if and only if a non-RosettaNet content is to be shipped in the payload portion of a RosettaNet Business Message.	Partners agree on this value. Refer to section 2.1.4.4 for more details.
Partner Identification (Known or Unknown Initiating Partner)	Identifies the trading partner who initiated this process by Global Business Identifier and optional Location ID.	
PIP Code	RosettaNet PIP Code of this message. Set by the initiating partner.	The valid value for this element MUST be obtained from the "PIPCode" element in the PIP Specification corresponding to the currently executing PIP.

Element Name	Description / Notes	Special Validation and Processing Rules
PIP Instance ID	The ID of this PIP instance.	MUST be unique within the context of the initiating partner.
PIP Version	RosettaNet PIP Version of this message. Set by the initiator of this transaction.	The valid value for this element MUST be obtained from the "PIPVersion" element in the PIP Specification corresponding to the currently executing PIP.~~. In the absence of such an element, the following two guidelines MUST be adhered to:~~ ~~1. For specifications that have been versioned according to the RosettaNet Technical Conventions and Style Guide, the value MUST be based on the version indicated on the title page of the specification document, and MUST be rewritten in the form 'CMM.mmS', where 'C' is the Category indicator (e.g. 'R' for Release, 'B' for Beta, 'V' for Validated). 'MM' is the Major version number. 'mm' is the minor version number. 'S' is the sequential letter that is incremented during a specification's Validation period, and is only present for approved but unvalidated specifications. The third set of numerals (Patch increment) are always dropped, as these increments have no effect on implementation. Examples are 'V01.03' or 'R02.00C'.~~ ~~2. For specifications that were versioned prior to the Style Guide, only the numeric portion of the version number, as found on the specification document's title page, is to be used (e.g. '1.2').~~
Process Control	Group of elements carrying information about the process within which this message is being sent.	
Proprietary Reference Identifier	A unique reference identifier for goods, services, or business documents.	Maximum length of 255.
Quality Of Service Element	Specifies a quality of service constraint item.	This element is specified for future backward compatibility. There are no valid values at this time. Receiver MUSTignore this element if set.

Element Name	Description / Notes	Special Validation and Processing Rules
Quality Of Service Specification	Specifies quality of service constraints for this message instance.	This element is specified for future backward compatibility. There are no valid values at this time. Receiver MUSTignore this element if set.
Quality Of Service Classification Code	Identifies the quality of service measurement category.	This element is specified for future backward compatibility. There are no valid values at this time. Receiver MUSTignore this element if set.
Service Content Control	Contains information about the Service Content.	
Signal Identity	The collection of properties that are used to identify a business signal.	
Standard Version (Action Identity)	The version of the standard with which the Service Content MUST be compliant.	MUST be set if and only if this is a non-RosettaNet-specified Service Content message.
To Role	The role the trading partner receiving this message plays in this PIP.	
To Service	The service to which this message is being sent.	
Uniform Resource Locator (Unknown Initiating Partner)	Specifies the URL to which replies MUST go in the case of an unknown body with whom a TPA MAY not exist.	If this is the first message in the PIP instance, MUST be specified if and only if the Partner Type is "Unknown" and the message is not requesting a synchronous response (see section 2.4). If Partner Type is "Unknown" and this value is not specified, further processing might not be possible.
Universal Resource Identifier (Attribute of Attachment Details)	Reference to the content ID of the attached document.	This value MUST follow the Content-ID reference syntax per RFC 2111 and MUST refer to the MIME Content-ID of the attachment.
Unknown Initiating Partner	An unknown partner initiating this PIP instance soliciting some public information through the RosettaNet PIP framework.	
Value	Identifies the quality of service measurement constraint.	Valid values are defined within the context of the Quality of Service Classification Code.

Element Name	Description / Notes	Special Validation and Processing Rules
Version Identifier (Signal Identity)	Identifies the version of the business signal that is carried in the Service Content.	The value for this element MUST be obtained from the Signal Version Identifier field of the identified Business Signal's Message Guideline.~~The valid value for this element MUST be obtained from the specification of the identified Business Signal. The value MUST be based on the version indicated in the signal's specification document, and MUST be rewritten in the form 'CMM.mmS', where 'C' is the Category indicator (e.g. 'R' for Release, 'B' for Beta, 'V' for Validated). 'MM' is the Major version number. 'mm' is the minor version number. 'S' is the sequential letter that is incremented during a specification's Validation period, and is only present for approved but unvalidated specifications. The third set of numerals (Patch increment), if present, is always dropped, as these increments have no effect on implementation. Examples are 'V02.00' or 'R02.01C'.~~

Example 3. Service Header Instance (Using PIP 3A4)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceHeader SYSTEM "ServiceHeader_MS_BV02_00.dtd">
<ServiceHeader>
<ProcessControl>
   <ActivityControl>
      <BusinessActivityIdentifier>Create Purchase
Order</BusinessActivityIdentifier>
      <MessageControl>
         <fromRole>
            <GlobalPartnerRoleClassificationCode>Buyer</GlobalPartner
RoleClassificationCode>
         </fromRole>
         <fromService>
            <GlobalBusinessServiceCode>Buyer
Service</GlobalBusinessServiceCode>
         </fromService>
         <Manifest>
            <Attachment>
               <description>
                  <FreeFormText>PDF version of PO</FreeFormText>
               </description>

<GlobalMimeTypeQualifierCode>PDFapplication/pdf</GlobalMimeType
QualifierCode>
               <UniversalResourceIdentifier>"cid:Attachment.
20001121T123000.000Z@this.example.com"</UniversalResourceIdentifier>
            </Attachment>
```



```
            <numberOfAttachments>
               <CountableAmount>1</CountableAmount>
            </numberOfAttachments>
            <ServiceContentControl>
               <ActionIdentity>
                  <GlobalBusinessActionCode>Purchase Order Request
Action</GlobalBusinessActionCode>
                  <VersionIdentifier>01.02</VersionIdentifier>
               </ActionIdentity>
            </ServiceContentControl>
         </Manifest>
         <toRole>
            <GlobalPartnerRoleClassificationCode>Seller</GlobalPartner
RoleClassificationCode>
         </toRole>
         <toService>
            <GlobalBusinessServiceCode>Seller
Service</GlobalBusinessServiceCode>
         </toService>
      </MessageControl>
   </ActivityControl>
   <GlobalUsageCode>Production</GlobalUsageCode>
   <KnownInitiatingPartner>
      <PartnerIdentification>
         <domain>
            <FreeFormText>DUNS</FreeFormText>
         </domain>
         <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      </PartnerIdentification>
   </KnownInitiatingPartner>
   <pipCode>
      <GlobalProcessIndicatorCode>3A4</GlobalProcessIndicatorCode>
   </pipCode>
   <pipInstanceId>
      <InstanceIdentifier>121212</InstanceIdentifier>
   </pipInstanceId>
   <pipVersion>
      <VersionIdentifier>01.02</VersionIdentifier>
   </pipVersion>
   <KnownInitiatingPartner>
      <PartnerIdentification>
         <domain>
            <FreeFormText>DUNS</FreeFormText>
         </domain>
         <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      </PartnerIdentification>
   </KnownInitiatingPartner>
</ProcessControl>
</ServiceHeader>
```

VERSIONING NOTES

RNIF 2.0 invalidates the 1.1 version of the Service Header. The new version to use is version 2.0 of the Service Header, which follows the Service Header DTD structure.

COMPLIANCE SUMMARY

This summary is for convenience only and is not guaranteed to contain all compliance statements. For complete compliance knowledge, read the entire specification.

A message that is compliant with RNIF 2.0 MUST have an XML document called Service Header. This document MUST have been packaged according to the packaging rules specified in section 2.3. The document MUST conform to the DTD defined above and MUST have values in conformance to the applicable Element Description table.

2.1.4 Payload Components

The payload part of the RosettaNet Business Message comprises the Service Content (which is either an action message or a signal message) and zero or more OPTIONAL attachments.

The payload is the actual business content that the Service Header describes or identifies. The Service Header format is fixed and independent of payload. The Service Content part of the payload (i.e., the action message or signal message) changes based on the specific business content being exchanged, which depends on the PIP type and instance. The attachments are also dynamic per instance of the business message as should be expected.

VERSIONING NOTES

"Payload" as a concept is new to RNIF 2.0, as are attachments. The RosettaNet Service Content is the same as in RNIF 1.1, except that in RNIF 2.0 it can contain non-RosettaNet content.

2.1.4.1 Service Content

The Service Content part of the payload contains business content that is in XML format. The Service Content is always either an action message or a signal message. The DTDs for all signal messages are specified by RosettaNet. The DTDs for PIP action messages MAY be specified by RosettaNet or by other standards bodies that have been sanctioned by RosettaNet.

PIPs must identify which are the allowed standards body(ies) that can supply content in the given PIP.

2.1.4.2 Handling Attachments

Payloads containing action messages could contain attachments. These attachments are typically supporting documents that accompany the business documents. Attachments need not be XML documents; some examples of attachments include: Word documents, GIF images, PDF files, TIF images, etc. Each attachment constitutes a separate MIME body part in the RosettaNet Business Message and MUST have the MIME Content-ID attribute specified (see section 2.3 for details). The Content-ID value for the attachment is also listed in the Service Header's Manifest element.

2.1.4.3 Referring to Attachments from within Service Content

As mentioned above, attachments to Service Content are sent as separate MIME body parts in the same RosettaNet Business Message. This method packages and ships the business content and attachments together. However, RosettaNet recognizes that it sometimes would be necessary to refer to attachments from within the Service Content. Since action messages (specified by RosettaNet or otherwise) are defined independently of the RosettaNet Implementation Framework, RNIF 2.0 defines a standard mechanism to refer to attachments from within XML Service Content documents and leaves it up to the Service Content DTD developers to make use of this mechanism.

Each attachment MUST be identified by the MIME header "Content-ID" in the RosettaNet Business Message. All XML elements that could refer to attachments MUST have the attribute "href" defined as one of the attributes for the XML element.

For example:

```
<!ELEMENT AnyElement (#PCDATA)>
<!ATTLIST AnyElement
  %miscAttributes;
 href CDATA #implied)>
```

An instance of the element "AnyElement" could then refer to the attachment as follows:

```
<AnyElement href="cid:<cid-of-attachment>"> ...
</AnyElement>
```

where <cid-of-attachment> is the value of Content-ID MIME header for the attachment.

For example, if the MIME part packaging of an attachment in a RosettaNet message occurs as follows:

```
--RN-Outer-Boundary—
Content-Type: image/gif
Content-Transfer-Encoding: Base64
Content-ID: <00180792811xyz@xyz.rosettanet.org>

[Attachment data goes here]

--RN-Outer-Boundary—
```

then an instance of the element "AnyElement" could refer to the attachment as follows:

```
<AnyElement href="cid:00180792811xyz@xyz.rosettanet.org">
</AnyElement>
```

COMPLIANCE SUMMARY

This summary is for convenience only and is not guaranteed to contain all compliance statements. For complete compliance knowledge, read the entire specification.

The MIME Content-ID attribute MUST be specified for all attachments.

The format cid:<value> MUST be used for the value of the href attribute.

Multiple elements MAY refer to the same attachment.

2.1.4.4 Shipping Non-RosettaNet Service Content in the Payload

A RosettaNet PIP definition, among such other things as activity names, actions, timeouts, and retry definitions, includes document type definitions and message guidelines for all the action messages in the PIP. Some Supply Chain Partners have expressed the need to use Document Type Definitions from other standards within a RosettaNet PIP. As a result, RNIF 2.0 specifies a mechanism to enable implementations to exchange non-RosettaNet Service Content within a RosettaNet Business Message.

Note, however, that these Document Type Definitions and versions MUST be sanctioned by RosettaNet (on a per-PIP basis). When such service content is allowed as an alternative to RosettaNet-provided Service Content, then trading partners need to decide in advance whether to use it.

If two trading partners decide to use non-RosettaNet Service Content, they MUST NOT alter anything in the PIP specification itself. They can only agree upon what Document Types Definitions and versions to use for all the action messages in the PIP. For instance, assume that Trading Partner X and Trading Partner Y decide to use the business message structures defined by the ABC standard for the High Tech Manufacturing industry, where ABC is a message exchange standard and does not deal with business process definitions. In such a case, the two trading partners need to agree on a common "ID" to bind this payload structure with the PIP version they execute. They MUST agree among themselves as to which RosettaNet-sanctioned message type and version MUST be used for the request and which message type and version MUST be used for the response. Let us assume that they choose to use ABC standard's DTD structures in order to execute the Purchase Order Management PIP and specifically, the PO version 1 for the request and a PO Acceptance version 1 for the response. They will then need to identify this "flavor" of their PIP with a unique identity, say "XY". This value "XY" will be used in the Partner-Defined PIP Payload Binding ID element in the Service Header.

The Partner-Defined PIP Payload Binding ID MUST be unique per the set of Trading Partners using it (therefore Message Standard and Standard Version can be inferred from it). This element MUST be set if ~~and only if~~ the PIP is executed in such a scenario. This element MUST NOT be set if the PIP is compliant with the regular RosettaNet PIP. Note that the combination of a PIP Code, PIP Version and Partner-Defined PIP Payload Binding ID identifies a unique set of Service Content types within the partners' systems

RosettaNet is not responsible for the maintenance of these non-RosettaNet DTDs.

2.2 Security Provisions and Trading Partner Authentication

This section specifies how S/MIME is used within RosettaNet for securing messages. It also establishes the norms RECOMMENDED by RosettaNet for use of digital signatures.

2.2.1 Use of S/MIME within RosettaNet

The use of S/MIME (Secure/Multipurpose Internet Mail Extensions) in RosettaNet is governed by IETF RFC 2311 "S/MIME Version 2 Message Specification" which describes the S/MIME v.2 format. RNIF 2.0 makes use of the enveloped and signed data types defined in the S/MIME specification.

S/MIME provides one format for enveloped-only data and several formats for signed-only data. RNIF 2.0 utilizes the enveloped and the multipart signed S/MIME formats. (See examples below for use of actual headers.)

A single procedure is used for creating MIME entities that are to be signed or enveloped. Some additional steps are RECOMMENDED to defend against known corruption that can occur during mail transport and that are of particular importance for clear-signing using the multipart/signed format. The rules for creating MIME entities for signing and enveloping are outlined in RFC 2311 and are defined in RFC 2045 – 2049.

According to S/MIME guidelines each MIME entity MUST be converted to a canonical form that can be uniquely and unambiguously represented in the environment where the signature is created and in the environment where the signature is verified. MIME entities MUST be presented in a canonical format for enveloping as well as signing. The S/MIME specification also recommends that entities such as 8-bit text and binary data be encoded with quoted-printable or base-64 transfer encoding. For this reason, all recipients MUST be able to read both quoted-printable and base-64 encoded messages.

The application/pkcs7-mime defined by S/MIME type carries PKCS #7 objects of several types, including envelopedData and signedData. The PKCS #7 object MUST always be BER encoding of the ASN.1 syntax describing the object. According to the S/MIME guidelines the contentInfo field of the carried PKCS #7 object MUST never be empty. Since PKCS #7 objects are binary data, in most cases base-64 or quoted printable transfer encoding is appropriate, in particular when used with SMTP transport. The transfer encoding used depends on the transport through which the object is to be sent, and is not a characteristic of the MIME type.

RNIF 2.0 uses S/MIME enveloped messages to secure parts ~~of the~~ of the RosettaNet business messages. The S/MIME specification recommends the following three-step process for creating enveloped messages:

1. The MIME entity is prepared for enveloping.

2. The MIME entity and other required data are processed into a PKCS #7 object of type envelopedData. The PKCS #7 object is inserted into an application/pkcs7-mime MIME entity.

3. Appropriate transfer encoding is applied to the parts of the MIME entity.

The smime-type parameter for enveloped-only messages is "enveloped-data". The file extension for this type of message is ".p7m".

Example 4. S/MIME Enveloped Message

```
Content-Type: application/pkcs7-mime;
     smime-type=enveloped-data;
     name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

RNIF 2.0 utilizes the multipart/signed form of the signed messages specified by the S/MIME specification. The S/MIME specification provides the following five-step process for creating multipart/signed messages:

1. The MIME entity is prepared for signing.

2. The MIME entity is presented to PKCS #7 processing in order to obtain an object of type signedData with an empty contentInfo field.

3. The MIME entity is inserted into the first part of a multipart/signed message.

4. Transfer encoding is applied to the detached signature obtained in step 2 and it is inserted into a MIME entity of type application/pkcs7-signature.

5. The MIME entity of the application/pkcs7-signature is inserted into the second part of the multipart/signed entity.

The multipart/signed Content type has two required parameters: the protocol parameter and the micalg parameter. For this MIME part the protocol parameter is "application/pkcs7-signature". The value of the micalg parameter is dependent on the message digest algorithm used in the calculation of the Message Integrity Check.

Example 5. S/MIME multipart/signed Message

```
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
```



```
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

2.2.2 Use of Digital Certificates within RosettaNet

RNIF 2.0 RECOMMENDS the use of digital certificates. Digital certificates are delivered as a part of the application/pkcs7-signature part of the multipart-signed RosettaNet message. RosettaNet uses RFC 2312 “S/MIME Version 2 Certificate Handling” as a guideline for use of digital certificates in RosettaNet messages. Due to the complexity of the certification process and overall immaturity of the existing PKI deployments, RosettaNet is much more tolerant in respect of the content of the certificates. This section establishes the norms RECOMMENDED by RosettaNet. The exact implementation of the certificate handling procedures and authentication semantics of the information in the digital certificate received with a RosettaNet message is left to the Trading Partner Agreement.

According to the S/MIME certificate handling specification, receiving agents MUST support X.509 v1 and X.509 v3 certificates. The specification also requires that end-entity certificates include an Internet mail address for the sender. Since RNIF 2.0 is defined in a transport-independent fashion, the Internet email address of the sender in the end-entity certificates MAY be omitted.

RNIF 2.0 aligns with the S/MIME certificate handling specification in that receiving agents MUST be able to handle an arbitrary number of certificates of arbitrary relationship to the message sender and to each other in arbitrary order. RNIF 2.0 also aligns with the S/MIME specification in the use of a single or a dual key pair for data signing and encryption: the choice of the number of the key pairs is left for the Trading Partner Agreement.

RNIF 2.0 requires that the sender MUST include any certificates that contain the signer's public key(s). The sender MAY include the associated issuer certificates. This measure allows establishing a simple and efficient way of associating the message sender with a particular Trading Partner profile.

RNIF 2.0 leaves it to the Trading Partner Agreement to determine the format of the certificate chains leading to the self-signed root Certificate Authority (CA) certificates. The recipient SHOULD be able to support the types of certificate chains (complete and incomplete) described in the S/MIME certificate handling specification and directly trusted certificates (empty certificate chain). All trust decisions are left to the Trading Partner Agreement. In full conformance with the S/MIME certificate handling specification, RosettaNet message recipients MUST support certificate chaining based on the distinguished name fields in the certificates. RNIF 2.0 REQUIRES verification of the signer’s certificate validity.

The X.509 v3 standard describes an extensible framework in which the basic certificate information can be extended and how such extensions can be used to

control the process of issuing and validating certificates. At present, there is no single, coherent view regarding which certificate extensions must be present in the X.509 v.3 digital certificates. RNIF 2.0 leaves the use of the particular X.509 v.3 certificate extensions to the Trading Partner Agreement. RNIF 2.0 also lessens the requirements of the S/MIME certificate handling specification and does not require the recipients to handle the subset of the certificate extensions listed in RFC 2312. RNIF 2.0 REQUIRES the recipient to abandon verification of messages that contain certificates with critical extensions that the recipient is unable to handle. It is RECOMMENDED that the UNP.MESG.SIGNERR event SHOULD be handled according to internal policies.

RNIF 2.0 RECOMMENDS but does not require the recipient to implement a certificate-revocation list (CRL) retrieval mechanism in order to gain access to certificate revocation information when validating certificate chains. RNIF 2.0 RECOMMENDS but does not require the recipient to retrieve and utilize CRL information every time a certificate is verified as part of a certificate chain validation, even if the certificate was already verified in the past. RNIF 2.0 does not specify which technique is used to validate certificates (e.g., via CRL, using the OCSP protocol, etc.). All certificate validation procedures are executed according to local security policy. RNIF 2.0 RECOMMENDS that the use of CRL information MAY be dictated by the value of the information that is protected.

2.3 RosettaNet Business Message Packaging and Unpackaging

This section specifies how the sender of the message assembles the defined message components and how the recipient extracts those components. It includes details on packaging and unpackaging RosettaNet Business Messages that have been encrypted and/or signed, as well as "plain" messages.

A RosettaNet Business Message is a combination of the individual business message components packaged into a MIME message, with appropriate MIME headers. Signed and enveloped content types per the S/MIME specification are used to provide authentication, message integrity, privacy, data security, and non-repudiation of origin. (See RFC 2311 for details.) Non-repudiation of receipt is achieved by signed Receipt Acknowledgments, which contain the digest of the received message.

RosettaNet Business Message packaging involves packaging the various business message components described in section 2.1 into MIME and/or S/MIME entities. Unpackaging involves extracting individual RosettaNet Business Message components from the MIME entities.

All packaging and unpackaging specifications within this section are independent of the transfer protocol used. However, some transfer protocols might not be able to handle binary or 8-bit data. Where one of these transfer protocols is used, content transfer encoding such as base-64 MUST be used to transform the binary and 8-bit data ~~to~~ into 7-bit encoding. Transfer protocol-specific bindings and transfer protocol headers are treated in the "RosettaNet Business Message Transfer" section.

The RosettaNet packaging specification follows standard MIME conventions, unless otherwise stated.

Note: Per standard MIME convention, MIME header names and values and parameter names ~~and values~~ are not case sensitive, while parameter values are normally case sensitive. The order of MIME headers in a part and the order of parameters in a header (if more than one is present) are also not significant. Additionally, values for MIME boundaries shown in the examples are just examples and SHOULD NOT be used as the actual values.

2.3.1 Definitions of Terms

This subsection describes the terms used throughout this section to refer to certain logical groups of the business message components. Note that these definitions are only logical because they do not include the extra entities included by MIME packaging such as the MIME headers themselves and the MIME boundaries.

These definitions use Backus Naur Form (BNF) for description.

Service Content: comprises an action message or a signal message.
Grammar Rule: Service-Content := Action-Message | Signal-Message

Attachments: Documents or files that are not part of the Service Content but need to be packaged and sent as a part of the RosettaNet Business Message.
Grammar Rule: Attachments := *Attachment

Payload: This refers to a logical group containing the *Service Content* and the *Attachments* (if any).
Grammar Rule: Payload := Service-Content Attachments

Payload Container: This term refers to a logical group containing the *Payload* and the Service Header.
Grammar Rule: Payload-Container := Service-Header Payload

RosettaNet Business Message: This term refers to a logical grouping of the *Payload Container,* the Delivery Header, and the Preamble. Note: A RosettaNet Business Message is sometimes referred to as "Business Message" for convenience.
Grammar Rule: RosettaNet-Business-Message := Preamble Delivery-Header Payload-Container

2.3.2 Using Intermediaries

Care has been taken to ensure that the use of an intermediary by a partner is kept as "transparent" as possible to the other partner. The idea is to enable transmitting messages through intermediaries without having to alter the message structure or perform heavy processing.

Hence, the packaging or unpackaging rules to be followed when an intermediary is involved are no different from those followed when the intermediary is not involved.

2.3.3 Packaging the RosettaNet Business Message

The RosettaNet Business message consists of the following components:

1. Preamble
2. Delivery Header
3. Service Header
4. Service Content
5. Attachments (if any)

Packaging involves encapsulating these various components using the MIME specification and optionally encrypting and/or signing the appropriate portions.

NOTES ON SIGNING AND ENCRYPTING

The decision on whether to encrypt depends on the agreement between the trading partners involved, and other factors such as the sensitivity of the actual service content and attachments. In order to provide flexibility, RNIF 2.0 allows encryption of either the entire Payload Container or just the Service Content. The choice depends on what the two trading partners agree upon, which may ultimately depend upon whether an intermediary needs access to the Service Header and/or the sensitivity of the data in the Service Header.

In order to make implementations simple, RNIF 2.0 only allows signing of the RosettaNet Business Message as a whole. In other words, RNIF 2.0 does not allow signing of individual or selective parts of the RosettaNet Business Message.

To protect sensitive information contained within a RosettaNet Exception Business Signal, if the message to which it is a response was encrypted and/or signed, the ~~an~~ Exception Business Message MUST likewise be encrypted and/or signed ~~in the same manner as the message to which it is a response~~.

PACKAGING NON-ROSETTANET CONTENT

As described earlier in section 2.1, action messages could be in a format defined by RosettaNet or any other standards body that is permitted by RosettaNet. The XML Service Header elements MUST clearly identify the nature of the Service Content. Refer to the description of Service Header for complete details.

GENERAL PACKAGING RULES

In encapsulating the components into a MIME entity, all body parts carrying only XML data MUST use the content type of application/xml and MAY be content-transfer-encoded (see RFC 2376). Also, all body parts MUST contain a Content-ID header. RNIF 2.0 REQUIRES this header for all MIME parts even though this header is optional according to the MIME specification (see RFC 2045). Additionally the Content-Location header defined in RFC 2557 MUST be used to label Preamble, Service Header, and Service Content parts. Use of this header to tag these parts allows the receiving entity to identify and perform any special handling of these elements. The values that MUST be used for the Content-Location header for the respective parts are specified in Table 4.

Table 4. Content Location Values

Body Part Carrying	**Content-Location Value** (case insensitive)
Preamble	RN-Preamble
Delivery Header	RN-Delivery-Header
Service Header	RN-Service-Header
Service Content	RN-Service-Content

The packaging specification uses the multipart/related MIME structure (see RFC 2387) to package plain components of the message. It also uses the S/MIME types multipart/signed and "application/pkcs7-mime" type with "smime-type=enveloped-data" for signing and enveloping content, respectively.

For the multipart/related content-type, the “type” parameter is mandatory and MUST be specified with a value corresponding to the “root” part of the multipart/related message (see below for more details). The “start” parameter is OPTIONAL and if present MUST contain the Content-ID value corresponding to the root part that is identified in the “type” parameter.

NOTES ON CONTENT TRANSFER ENCODING

When deciding on a particular content-transfer-encoding to apply to a MIME entity (multipart section), consideration SHOULD be given to the characteristics of the data content of that entity, as well as of the transfer mechanism over which the message will be carried. In general, if it is known for certain (through an agreement between the trading partners) that the entire communication path allows binary data to be carried, then it is most efficient to use binary encoding (no transformation) for all MIME multipart entities. If this assumption cannot be made (for example, if the delivery mechanism is determined after packaging, or if an intermediary may route the message using unknown protocols), then all MIME entities that are not already compliant with 7bit encoding MUST be transformed by applying either quoted-printable or base64 transfer encoding. Refer to RFC 2045 for further details about the Content-Transfer-Encoding MIME header.

PACKAGING STEPS

This section describes the steps necessary to package the RosettaNet Business Message. These steps are descriptive rather than prescriptive. The implementer MAY use any procedure or sequence to package a message as long as the result is the same.

The Service Content and the Attachments, if any, are created as per the PIP specification.

The Service Header is created using information about the PIP being executed, the Service Content, and the Attachments, if any.

The Delivery Header is created using such information as the sender identification, the receiver identification, and a ~~globally unique~~ message tracking ID.

The Preamble is created as per the Preamble specification.

Once these components are created, packaging of the RosettaNet Business Message commences. The selection and flow of packaging steps varies depending upon whether the RosettaNet Business Message is to be encrypted or not, and upon whether it is to be signed or not. If the message is not to be encrypted, the steps in "Packaging without Encryption" MUST be performed. If the message is to be encrypted, the steps in "Packaging with Encryption" MUST be performed. Finally, if the message is to be signed, the steps in "Signing the Package" MUST be performed.

PACKAGING WITHOUT ENCRYPTION

If encryption is not required, the Preamble, the Delivery Header, the Service Header, the Service Content and the attachments (if any) are packaged into a multipart/related message (see RFC 2387). Although the Content-ID header is optional in MIME, RosettaNet requires that each of the body parts of the multipart/related message contains the Content-ID header as previously described. Note that the values in the Content-ID header MUST be globally unique (see RFC 2045). Additionally, the Preamble, the Delivery Header, Service-Header, and Service-Content MUST also have the Content-Location header with the respective values "RN-Preamble", "RN-Delivery-Header", "RN-Service-Header", and "RN-Service-Content".

In creating this multipart message, the Preamble MUST be the first body part, the Delivery Header the second body part, the Service Header third, and the Service Content the fourth body part. Attachments (if any) appear from the fifth body part onwards. There is no specific order in which these attachments are arranged; however, the order in which the attachments are listed in the manifest part of the Service Header MAY be followed for convenience.

The mandatory "type" parameter of the multipart/related content-type header MUST have the value "application/xml", corresponding to the Preamble (which happens to be the root or first part). The OPTIONAL "start" parameter, if present, MUST contain the Content-ID value of the Preamble.

This constitutes the entire (unencrypted) RosettaNet Business Message without a signature.

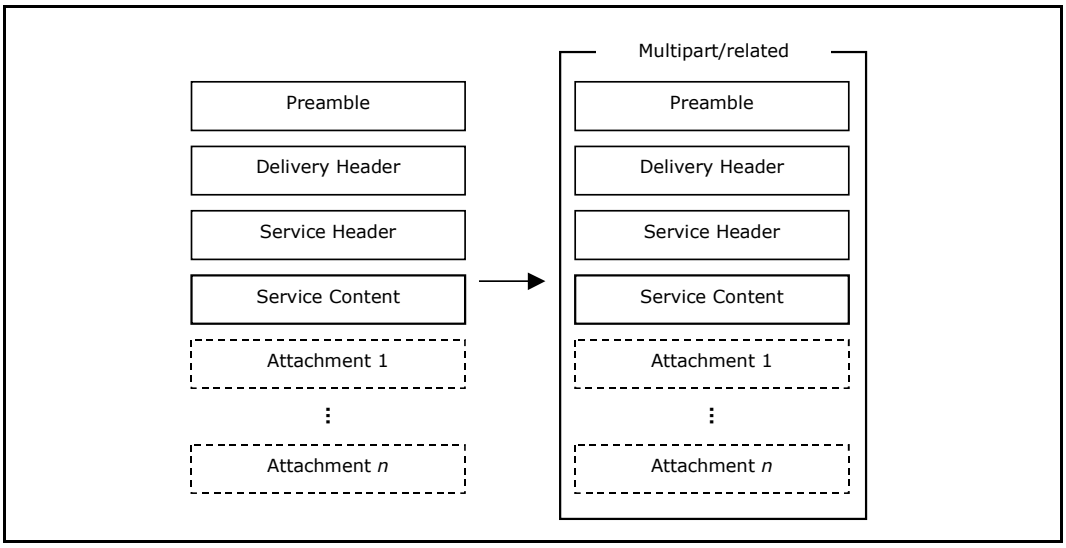


Figure 8. Packaging RosettaNet Business Message without Encryption

Example 6. Packaged RosettaNet Business Message without Encryption

```
Content-Type: multipart/related; boundary="RN-Outer-Boundary";
              type="application/xml"
Content-Description: This is the RosettaNet Business Message

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RN-Preamble"
Content-ID: <content-ID-for-Preamble>

[Preamble goes here]

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RN-Delivery-Header"
Content-ID: <content-ID-for-Delivery-Header>

[Delivery Header goes here]

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RN-Service-Header"
Content-Description: RosettaNet-Service-Header
Content-ID: <content-ID-for-Service-Header>

[Service Header goes here]

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Description: RosettaNet-Service-Content
Content-Location: "RN-Service-Content"
Content-ID: <content-ID-for-Service-Content>

[Service Content goes here]

--RN-Outer-Boundary
Content-Type: Image/jpeg
Content-Description: A Diagram of the product
Content-ID: diag-123-16776789.ghfg.efg-xcabc.071400
```

```
[Attachment 1 goes here]

--RN-Outer-Boundary
Content-Type: Image/tiff
Content-ID: diag-123456789.ghfg.efg-xcabc.08233

[Attachment 2 goes here]

--RN-Outer-Boundary--
```

PACKAGING WITH ENCRYPTION

The Service Header MAY either be encrypted along with the Service Content and Attachments or be left unencrypted while the Service Content and the Attachments (if any) alone are encrypted. Depending on which of these two options is used, the rules under "Encrypting the Entire Payload Container" or "Encrypting the Payload" are used respectively.

ENCRYPTING THE ENTIRE PAYLOAD CONTAINER

If encryption of the Service Header is required, the Service Header, the Service Content and the Attachments (if any) are packaged into a MIME multipart/related message (see RFC 2387). (This is the Payload Container.) Although the Content-ID header is optional in MIME, RosettaNet REQUIRES that each of the body parts of the multipart/related message contains the Content-ID header (see RFC 2045). Note that the values of the Content-ID header MUST be globally unique (see RFC 2045). Additionally, the Service Header and Service Content MUST also each have the Content-Location header with the values "RN-Service-Header" and "RN-Service-Content", respectively.

In creating this multipart/related message, the Service Header MUST be the first body part and the Service Content the second. Attachments (if any) appear from the third body part onwards. There is no specific order in which these attachments are arranged; however, the order in which the attachments are listed in the manifest part of the Service Header MAY be followed for convenience.

The mandatory "type" parameter of the multipart/related content-type header MUST have the value "application/xml", corresponding to the Service Header (which happens to be the root or first part). The OPTIONAL "start" parameter, if present, MUST contain the Content-ID value of the Service Header.

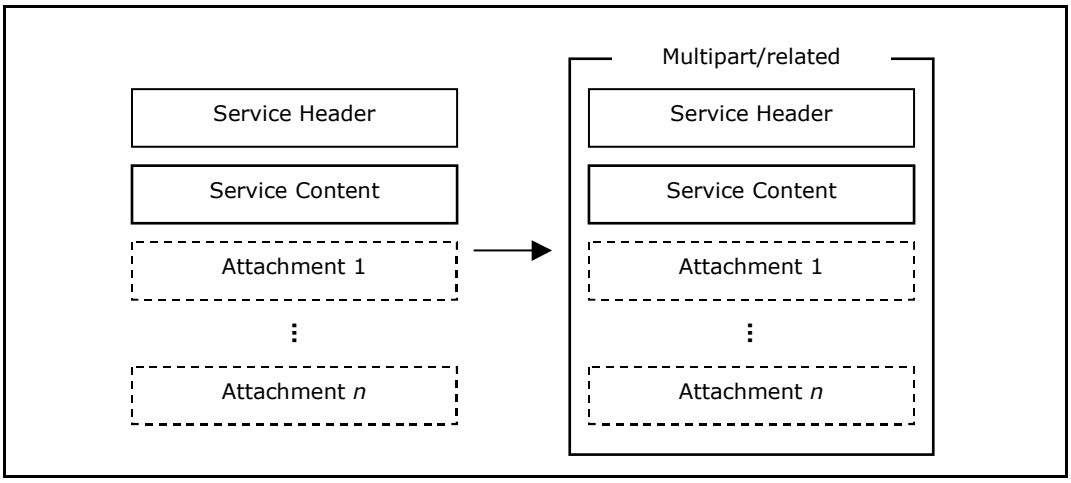


Figure 9. Packaging Payload Container Prior to Encryption

Example 7. Packaged Payload Container Prior to Encryption

```
Content-Type: multipart/related;
              boundary="RN-PayCnt-Boundary";
              type="application/XML";
              start="<content-ID-for-Service-Header>"
Content-Description: This is the payload container

--RN-PayCnt-Boundary
Content-Type: application/XML
Content-Description: RosettaNet-Service-Header
Content-Location: "RN-Service-Header"
Content-ID: <content-ID-for-Service-Header>

[Service Header goes here]

--RN-PayCnt-Boundary
Content-Type: application/XML;
Content-Description: RosettaNet-Service-Content
Content-Location: "RN-Service-Content"
Content-ID: <content-ID-for-Service-Content>

[Service Content goes here]

--RN-PayCnt-Boundary
Content-Type: Image/jpeg;
Content-Description: A Diagram of the product
Content-ID: diag-987654321.ghfg.efg-xcabc.00112233

[Attachment goes here]

--RN-PayCnt-Boundary--
```

The resulting multipart/related message is enveloped to create an S/MIME enveloped message using the "application/pkcs7-mime" content-type with "smime-type=enveloped-data" (see RFC 2311). RNIF 2.0 does not require any particular cipher strength or algorithm for data protection or encryption. These settings are retrieved from the Trading Partner Database as part of the Trading Partner Agreement and are ultimately determined by corporate policy, import and export restrictions, etc. (See RFC 2311 and also section 2.2.1 of this specification for complete details.)

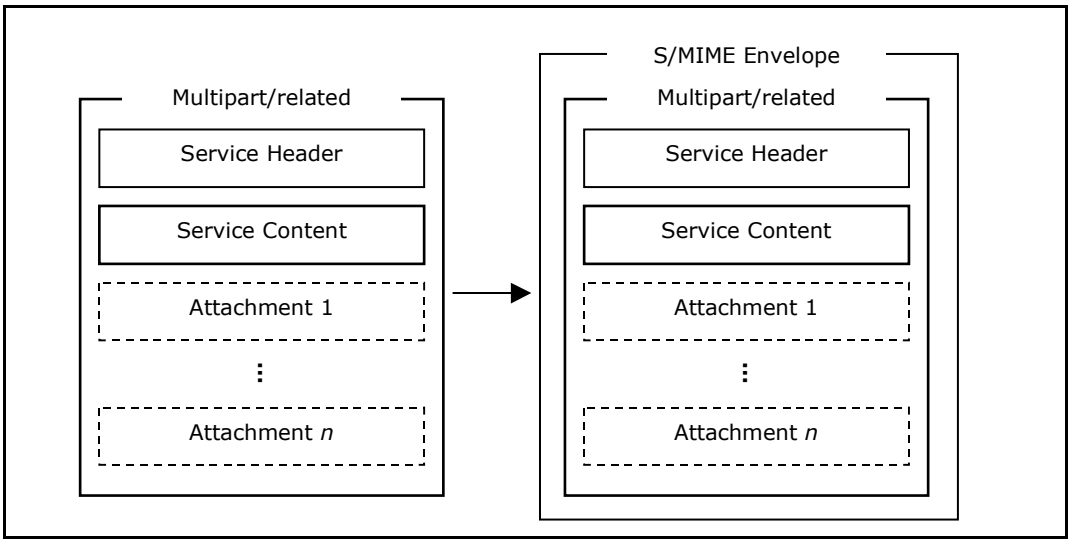


Figure 10. Encrypting the Payload Container

Example 8. Encrypted Payload Container

```
Content-Type: application/pkcs7-mime;
     smime-type=enveloped-data;
     name=something.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=something.p7m

[The Base64-encoded PKCS #7 object goes here]
```

In Example 8, the base64-encoded PKCS #7 object is the payload container packaged as a multipart/related message that was shown in Example 7. See RFC 2311 for details on how to create this object.

The Preamble, the Delivery Header, and the S/MIME enveloped message are then packaged into a multipart/related message with the Preamble as the first body part, the Delivery Header as the second, and the S/MIME entity as the third. The mandatory "type" parameter of the multipart/related content-type header MUST have the value "application/xml", corresponding to the Preamble (which happens to be the root or first part). The OPTIONAL "start" parameter, if present, MUST contain the Content-ID value of the Preamble.

The result of this packaging constitutes the entire encrypted RosettaNet Business Message without a signature in the case of the encrypted payload container.

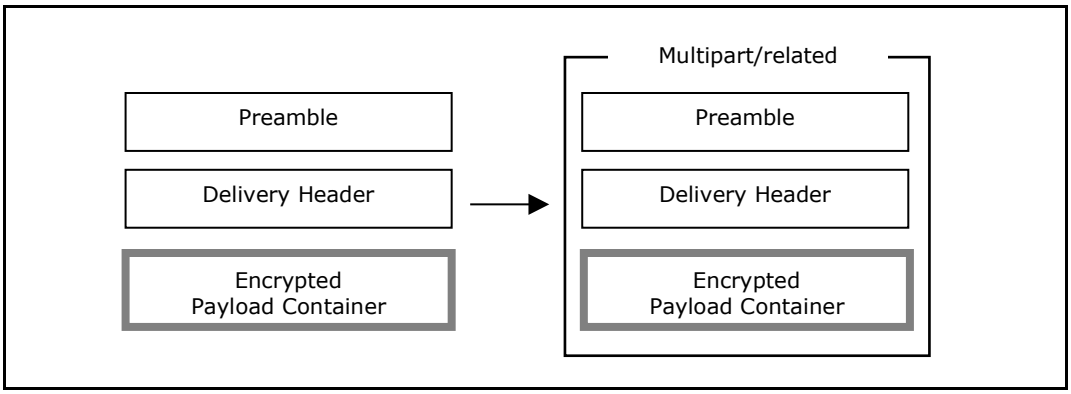


Figure 11. Packaging RosettaNet Message with Encrypted Payload Container

ENCRYPTING THE PAYLOAD

If the Service Header is required to be left unencrypted, the Service Content and the Attachments, if any, are packaged into a MIME multipart/related message (see RFC 2387). Although the Content-ID header is optional in MIME, RosettaNet REQUIRES that each of the body parts of the multipart/related message contain the Content-ID header (see RFC 2045). Note that the values of the Content-ID header MUST be globally unique (see RFC 2045). Additionally, the Service Header and Service Content MUST also each have the Content-Location header with the values "RN-Service-Header" and "RN-Service-Content", respectively.

In creating this multipart/related message, the Service Content MUST be the first body part and the Attachments, if any, MUST appear from the second body part onwards. There is no specific order in which these attachments are arranged; however, the order in which the attachments are listed in the manifest part of the Service Header MAY be followed for convenience.

The mandatory "type" parameter of the multipart/related content-type header MUST have the value "application/xml", corresponding to the Service Header (which happens to be the root or first part). The OPTIONAL "start" parameter, if present, MUST contain the Content-ID value of the Service Content.

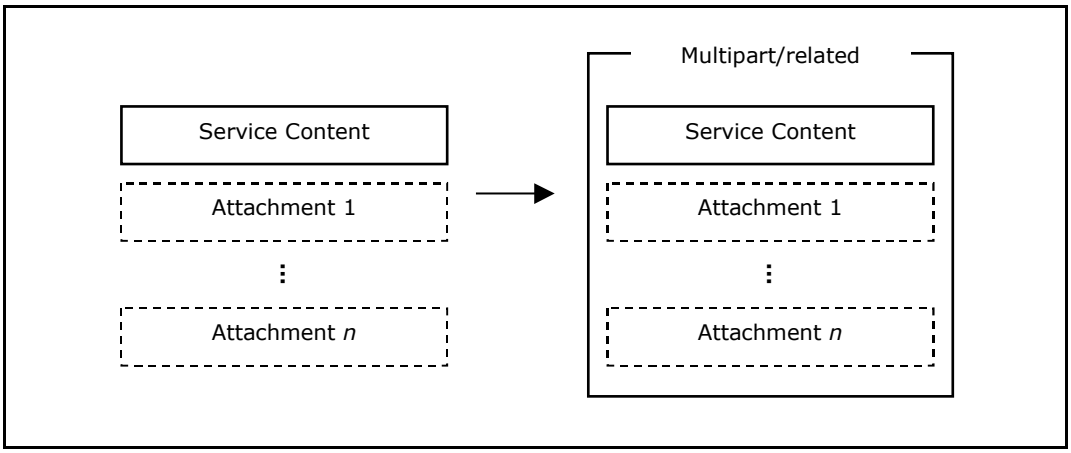


Figure 12. Packaging Payload Prior to Encryption

Example 9. Packaged Payload Prior to Encryption

```
Content-Type: multipart/related;
              boundary="RN-PayCnt-Boundary";
              type="application/XML";
              start="<content-ID-for-Service-Content>"
Content-Description: This is the payload

--RN-PayCnt-Boundary
Content-Type: application/XML;
Content-Description: RosettaNet-Service-Content
Content-Location: "RN-Service-Content"
Content-ID: <content-ID-for-Service-Content>

[Service Content goes here]

--RN-PayCnt-Boundary
Content-Type: Image/jpeg;
Content-Description: A Diagram of the product
Content-ID: diag-987654321.ghfg.efg-xcabc.00112233

[Attachment goes here]

--RN-PayCnt-Boundary--
```

The resulting multipart/related message is enveloped to create an S/MIME enveloped message using the "application/pkcs7-mime" content-type with "smime-type=enveloped-data" (see RFC 2311). RNIF 2.0 does not require any particular cipher strength or algorithm for data protection or encryption. These settings are retrieved from the Trading Partner Database as part of the Trading Partner Agreement and are ultimately determined by corporate policy, import and export restrictions, etc.

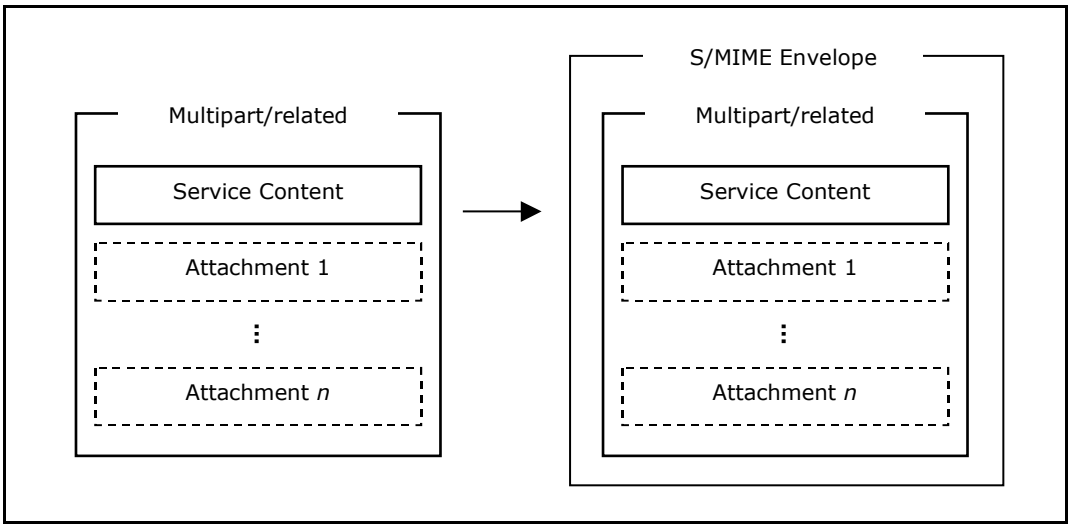


Figure 13. Encrypting the Payload

Example 10. Encrypted Payload

```
Content-Type: application/pkcs7-mime;
     smime-type=enveloped-data;
     name=something.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=something.p7m

[The Base64-encoded PKCS #7 object goes here]
```

In Example 10, the base64-encoded PKCS #7 object is the payload packaged as a multipart/related message (as shown in Example 9). See RFC 2311 for details on how to create this object.

The Preamble, the Delivery Header, the Service Header, and the S/MIME enveloped message are then packaged into a multipart/related message with the Preamble as the first body part, the Delivery Header as the second, the Service Header as the third, and the S/MIME entity as the fourth. The mandatory “type” parameter of the multipart/related content-type header MUST have the value “application/XML”, corresponding to the Preamble (which happens to be the root or first part). The OPTIONAL “start” parameter, if present, MUST contain the Content-ID value of the Preamble.

The result of this packaging constitutes the entire encrypted RosettaNet Business Message without a signature in the case of the encrypted payload.

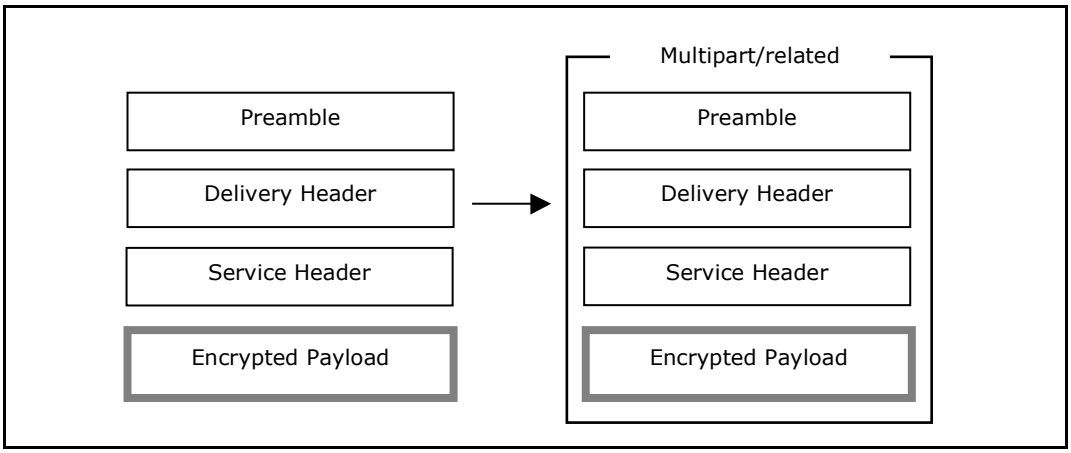


Figure 14. Packaging RosettaNet Message with Encrypted Payload

SIGNING THE ROSETTANET BUSINESS MESSAGE

If signature is required, the RosettaNet Business Message, whether encrypted or not, is signed following S/MIME conventions as specified in the "General Packaging Rules" section above. Specifically, the multipart/signed content type MUST be used for this purpose.

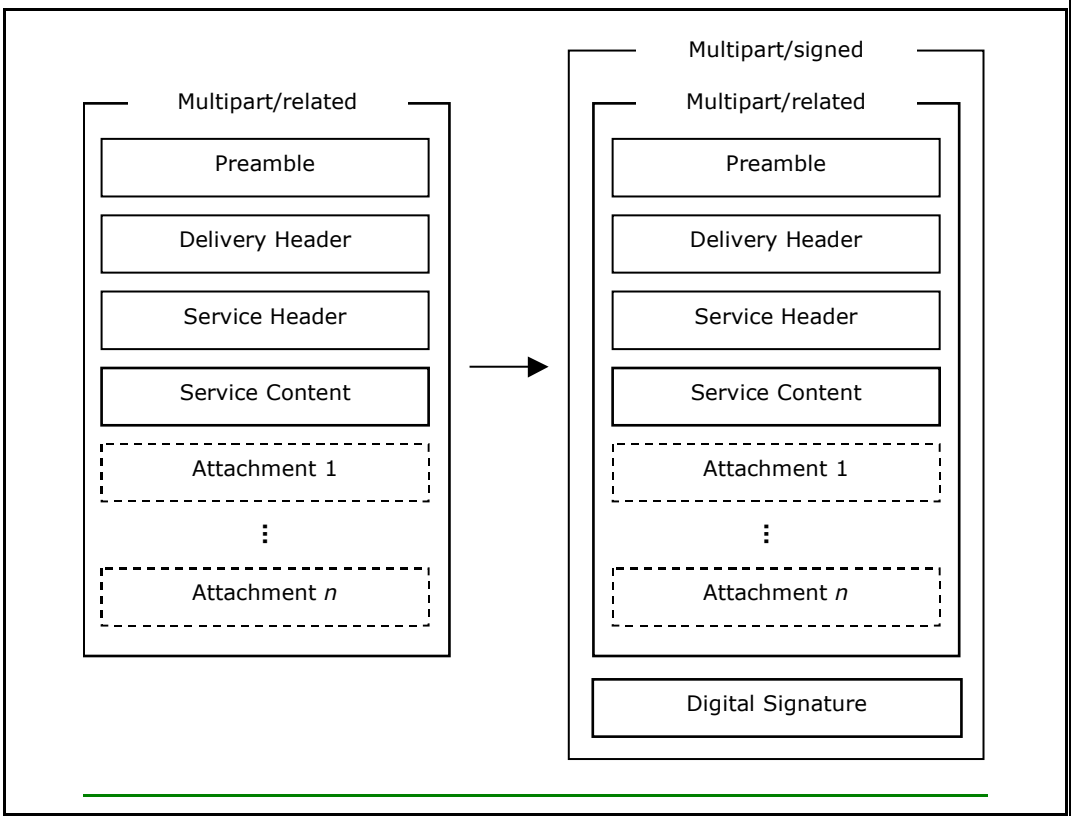


Figure 15. Signing the Unencrypted RosettaNet Business Message

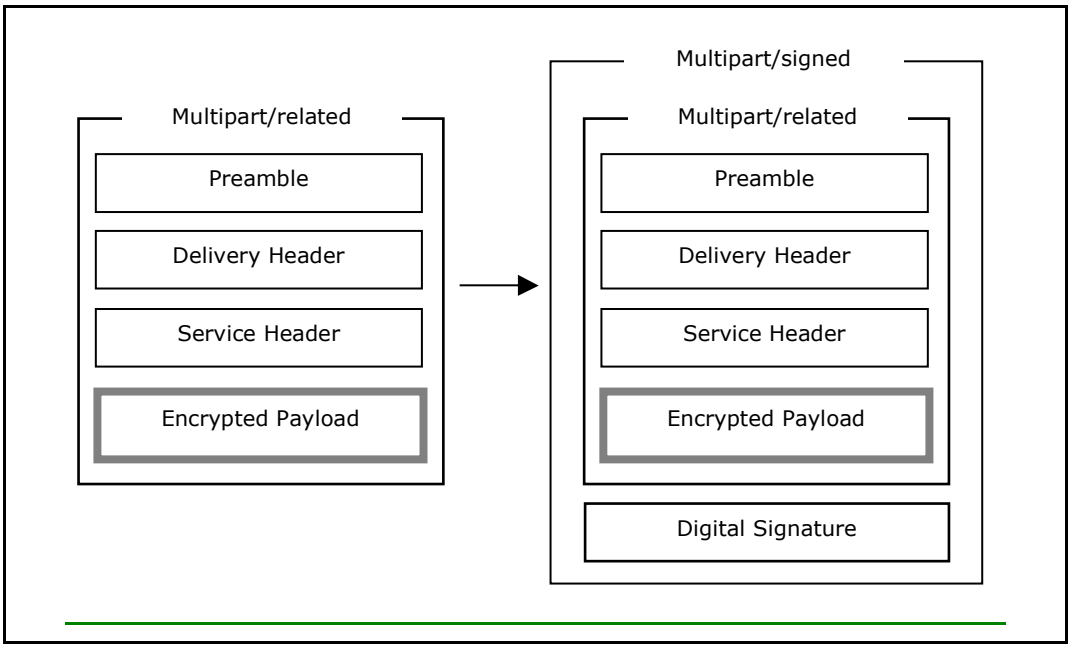


Figure 16. Signing the Encrypted RosettaNet Business Message (Payload Encrypted)

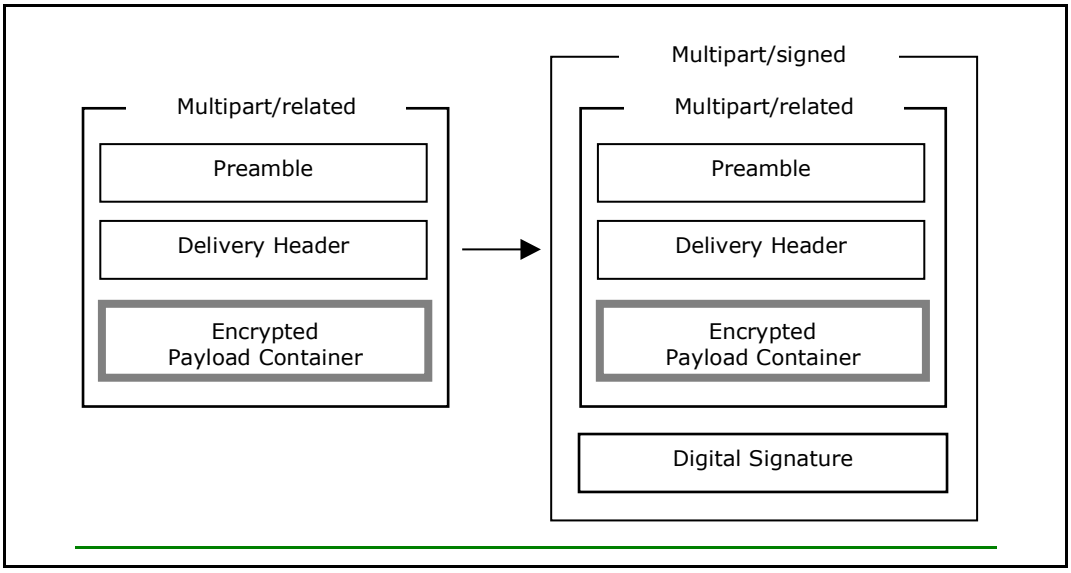


Figure 17. Signing the Encrypted RosettaNet Business Message (Payload Container Encrypted)

Example 11. Signed RosettaNet Business Message

```
Content-Type: multipart/signed;
              boundary="RN-Signature-Boundary";
              protocol="application/pkcs7-signature";
              micalg=sha1
Content-Description: This is a Signed RosettaNet Business Message

--RN-Signature-Boundary
```

```
[The RosettaNet Business Message to be signed goes here]

--RN-Signature-Boundary
Content-Type: Application/pkcs7-signature; name="detached.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
Content-Description: This is the signature for the Business Message

[The base64-encoded PKCS7 Detached Signature goes here]

--RN-Signature-Boundary--
```

REQUESTING SYNCHRONOUS RESPONSE

Note that above packaged message can now be transmitted via any transfer protocol. A detailed discussion on the transfer protocol specifics can be found in section 2.4. If the response for the message being so sent is required to be received synchronously, then the message MUST be sent via HTTP. In such a case, the HTTP entity header "x-RN-Response-Type" that indicates that the response be received synchronously MUST be specified. Refer to section 2.4 for more details on this header. Refer to section 2.6 for detailed rules on PIPs that can allow synchronous responses.

HANDLING PACKAGING ERRORS

Errors that are encountered during packaging are handled as follows:

- If the message being packaged is the first message in the PIP (i.e., the partner has never seen a message for this PIP instance before), then the error MAY be logged internally and/or handled according to the local policy.
- If the message being packaged is a response action message, unless the error is generic enough to happen while packaging any message — either fatal, irrecoverable, or both — then an Exception of type "General Exception" with error code PKG.MESG.GENERR (see Table 6~~Table 6~~~~Table 6~~~~Table 6~~~~Table 6~~~~Table 6~~~~Table 6~~~~Table 5~~) SHOULD be sent to the partner and the local instance of the process must be aborted. If t~~T~~he packaging/transmission of the exception ~~may~~ fails, ~~in which case~~ a Notification of Failure PIP instance SHOULD NOT be initiated.
- If the message being packaged is a signal, such as Receipt Acknowledgment or Exception, then the error MAY be logged internally and/or handled according to local policy. Also, the local instance of the PIP process MUST be terminated.

COMPLIANCE SUMMARY

This summary is for convenience only and is not guaranteed to contain all compliance statements. For complete compliance knowledge, read the entire specification.

The rules specified in the "General Packaging Rules" and "Packaging Steps" sections above MUST be followed.

2.3.4 Unpackaging the RosettaNet Business Message

This section of the specification discusses the unpackaging of the RosettaNet Business Message. Critical to the discussion of unpackaging is the handling of errors.

The Delivery Header carries the sender ID. The Service Header carries information on which PIP is being executed, as well as the Instance ID of the message. This information is necessary for the recipient of the message to notify the sender in case of errors in the message.

If the recipient encounters errors before successfully reading the Delivery Header and the Service Header, a mechanism is needed to identify the sender and other information so that errors can be reported back. This is accomplished through transport-level debug headers that supply this information. However, it is expected that debug headers would only be used in the set-up phases of new systems and/or when starting to implement RosettaNet PIPs with new trading partners. RosettaNet discourages the use of debug headers during production for obvious reasons. Refer to the sections on debug headers in section 2.4 for further details.

2.3.4.1 Unpackaging Steps

Unpackaging involves extracting the various components of the business message and simultaneously performing validation steps where applicable.

The steps described in this section are descriptive rather than prescriptive.

IDENTIFYING THE RESPONSE TYPE

In the case of a message received through an HTTP post, the requester posting the message may have requested that the response be sent back synchronously, on the same HTTP connection. This information is carried in the HTTP entity header "x-RN-Response-Type". Note that such a synchronous response is only possible if the requesting message came through HTTP. If the message was received through another transfer protocol, or if the above header is not present, then the message MUST be treated as if the response is to be sent asynchronously.

VERIFYING THE SIGNATURE

If the incoming RosettaNet Business Message is signed, the recipient MUST verify the signature. Signature verification and Sender Authentication are usually done together. Hence, in order to perform signature verification effectively, this step MAY be postponed until the Delivery Header is extracted completely. Refer to the section "Authenticating the Sender" for more details. The incoming message MUST be discarded if the signer is either unknown or not trusted, if the integrity of the message cannot be verified, or if this step failed for any other reason. In such cases, the error UNP.MESG.SIGNERR MAY be internally logged according to local policy. An Exception MUST NOT be sent to the sender of the message unless the transport headers carried debug information. If the message contained debug information in the transport headers, and if the recipient's policy allows notification of security errors to

the sender, an Exception MAY be sent. However, this is not recommended for security reasons.

EXTRACTING AND VALIDATING THE PREAMBLE

The Preamble, which is the first body part of the multipart/related message, is extracted and validated. For detailed rules on validation of any XML body part, refer to section 2.1.2.2.

If any of these tasks fail, the message MUST be discarded and the error UNP.PRMB.READERR or UNP.PRMB.VALERR (as the case may be) MAY be logged internally per local policy. An Exception signal MUST NOT be sent to the sender at this point as the sender is not yet identified, unless the incoming message contained debug headers in the transport headers. If the message contained debug headers in the transport headers, and if the recipient's policy allows notifying the sender of errors during the initial setup stages (debug stages), an Exception with type value of "General Exception" MAY be sent.

EXTRACTING THE DELIVERY HEADER

The second body part, which is the Delivery Header, is extracted and validated per the validation rules. If an error is encountered, then the message MUST be discarded and the error UNP.DHDR.READERR or UNP.DHDR.VALERR (as the case may be) MAY be logged internally per local policy. An Exception signal MUST NOT be sent to the sender at this point as the sender is not yet identified, unless the incoming message contained debug headers in the transport headers. If the message contained debug headers in the transport headers, and if the recipient's policy allows notifying the sender of errors during the initial setup stages (debug stages), an Exception with type value of "General Exception" MAY be sent.

AUTHENTICATING THE SENDER

Once the sender ID is extracted from the Delivery Header, the sender is authenticated as follows:

If the message was signed, verify that the signature belongs to the trading partner who sent this message. Authentication failures MAY be logged internally. An Exception MUST NOT be sent for security reasons. As in the other cases, if local policy allows, an Exception MAY be sent if the debug header is present in the transport headers in the incoming message.

Note that in the case of an unknown sender, the message will not be signed, and therefore no authentication will be needed.

EXTRACTING THE SERVICE HEADER

The third body part of the Multipart/related message is extracted.

If the content-type is "application/XML", then the Service Header was not encrypted. In such a case, this body part constitutes the Service Header.

If the content-type is "application/pkcs7-mime", then the payload and the Service Header were encrypted. In this case, this body part MUST be decrypted. Decryption MUST result in a multipart/related entity. The first body part in this enclosed multipart/related entity is extracted. This MUST be the Service Header.

If the decryption or the extraction of the Service Header fails, the message is discarded and the error UNP.MESG.DCRYPTERR or UNP.SHDR.READERR (as the case may be) MAY be logged internally, per local policy. As in previous cases, an Exception signal MUST NOT be sent to the sender at this point, as the ~~sender is not yet identified~~re is not yet enough information available to populate the Exception's Service Header, unless the incoming message contained debug headers in the transport headers. If the message contained debug headers in the transport headers, and if the recipient's policy allows notifying the sender of errors during the initial setup stages (debug stages), an Exception with type value of "General Exception"MAY be sent

VALIDATING THE SERVICE HEADER

The Service Header MUST be validated per the rules specified in section 2.1.2.2. If the header is found to be invalid (i.e., the error UNP.SHDR.VALERR applies), the error MAY be logged internally, per local policy. Again, an Exception signal cannot be sent to the sender at this point ~~as the sender is not yet identified~~, unless the incoming message contained debug headers in the transport headers. If the message contained debug headers ~~in the transport headers~~, and if the recipient's policy allows notifying the sender of errors during the initial setup stages (debug stages), an Exception MAY be sent.

Once the contents of the Service Header are extracted, the following validations MUST be performed:

- **Sequence validation.** The incoming message is related to the proper instance of an already executing PIP, or a new PIP instance is initiated if this is the first message of the PIP. If this step fails (e.g., if the message does not correspond to any PIP configured between the sender and the recipient, or if the instance IDs or the PIP/activity/action codes do not correspond to valid sequence (for instance, the request was referring to PIP 3A4, while the response says it is for PIP 3A7)), then an Exception MUST be sent to the sender if the incoming message is an action message. The exception type in the Exception is set to "General Exception" and the error code is set to UNP.MESG.SEQERR. If the incoming message is a signal, then the error MAY be logged according to local policy.

- **Synchronous Response Specification Verification.** If the incoming message is the first message for this PIP instance, and is received through an HTTP POST, and requires that the response be sent synchronously in the same HTTP connection, and the recipient supports synchronous message exchange, then the recipient MUST verify that the PIP specification allows for a synchronous response for this message. If such verification fails, then the error MAY be logged internally and an exception MUST be sent back synchronously, within the same HTTP connection. Similarly, if the HTTP header requires asynchronous response and the PIP specification prohibits asynchronous response, then an exception MUST be sent back asynchronously, if the action requires either a response or a Receipt Acknowledgment. If neither a response to this action nor a Receipt

Acknowledgment is required, then a Notification of Failure PIP is initiated. The error code used for this error, in case an Exception needs to be sent, is UNP.MESG.RESPTYPERR.

- **Authorization of Sender.** Note that though the sender's signature may have already been verified earlier while unpacking the Delivery Header, authorization of the sender (i.e., verifying whether the sender has the authority to participate in this PIP), cannot happen until the Service Header is unpacked. Any error in such verification is treated as a security error and MAY result in internal logging. An Exception MUST NOT be sent back to the sender of the message for security reasons. As in the other cases, if local policy allows, then an Exception MAY be sent if the debug header is present in the transport headers in the incoming message. However, if the incoming message requires synchronous response, failure to authenticate or authorize the sender MUST result in the receiver either sending an HTTP 403 response code or closing the connection with no response.

- **Manifest Verification.** If this is an action message, then the manifest is verified against the attachments (for the existence of the number of attachments as specified in the manifest, the existence of the specified Content-ID, and the corresponding content-type). If the verification fails, an Exception is sent to the sender with the exception type of "General Exception" and an error code of UNP.SHDR.MNFSTERR. This Exception MUST also be the result if the manifest indicates that Non-RosettaNet Service Content is present in the message, and such content is not supported by the solution. ~~-~~Note that the manifest ~~is~~ verification step MAY be deferred until the entire message is unpacked. However, this step MUST be performed before sending a Receipt Acknowledgment. The result of the verification MUST be the same whether this step is performed now or later.

EXTRACTING AND VALIDATING THE SERVICE CONTENT

The Service Content is extracted. (Note that whether or not the message was encrypted, either the Service Content or the encrypted Service Content is the body part after the Service Header.) In case the Service Content was encrypted, it MUST be decrypted. The Service Content is validated per the rules specified in section 2.1.2.2.

Processing an Action Message

If this is an action message, failure to~~-~~ decrypt, read or validate the Service Content MUST result in an Exception being sent ~~-~~if either a Receipt Acknowledgment or a response is required for this action. In such a case, the exception type is "Receipt Acknowledgment Exception". The error codes to use in the exception are UNP.MESG.DCRYPTERR, UNP.SCON.READERR, or UNP.SCON.VALERR, depending on whether the error happened while decrypting, reading or validation of the Service Content, respectively.

If neither Receipt Acknowledgment nor Response is required, then exceptions in processing the action message MUST result in initiation of the Notification of Failure PIP.

Refer to the below step "Processing Attachments" for more detailed instructions on extracting and validating the attachments.

If the processing of the action message completed without any error, the message MUST be persisted per local policy. A Receipt Acknowledgment MUST then be sent if the action requires a Receipt Acknowledgment. If the incoming message was signed and non-repudiation of receipt is required, then the Receipt Acknowledgment MUST carry the digest of the incoming message. For rules on computing the digest refer to the "Non-Repudiation of Receipt" section.

Processing a Signal Message

If this is an Exception, then the corresponding PIP instance must be aborted despite inability to read the Exception. If this is a Receipt Acknowledgment, then failure to read or validate the Receipt Acknowledgment MUST be treated similarly to the case where the Receipt Acknowledgment was never received; this error MAY also be logged internally. If the signal passes validation it is persisted per local policy. The corresponding PIP instance either completes (if this is the final signal) or continues (if this is not the final signal).

Processing Attachments

Body parts that follow the Service Content must be treated as attachments. Each attachment body part must specify the Content-ID for the attachment. If the Content-ID is missing or invalid, the receiver MUST send a General Exception back to the requester.

NON-REPUDIATION OF RECEIPT

When non-repudiation of receipt of an action message is required, the recipient of the message computes a digest of the received multipart/related body part, which is the first body part of the multipart/signed message. This computation MAY have been done as part of the signature validation step. The~~is~~ digest MUST ~~be extracted from the original (received) signed message,~~then be base-64 encoded (if not already), and included in the Receipt Acknowledgment in the "OriginalMessageDigest" field.

Note that non-repudiation of receipt is only required when the message is being accepted for processing. Hence, for messages that result in an Exception while unpackaging or validation, there is no need for non-repudiation of receipt.

UNPACKAGING AND ERROR HANDLING SUMMARY

This section summarizes in graphical and tabular form the entire message processing flow and the error handling processes and messages discussed in the previous sections.

2.6.7 shows the mandatory Error Codes and the associated descriptions.

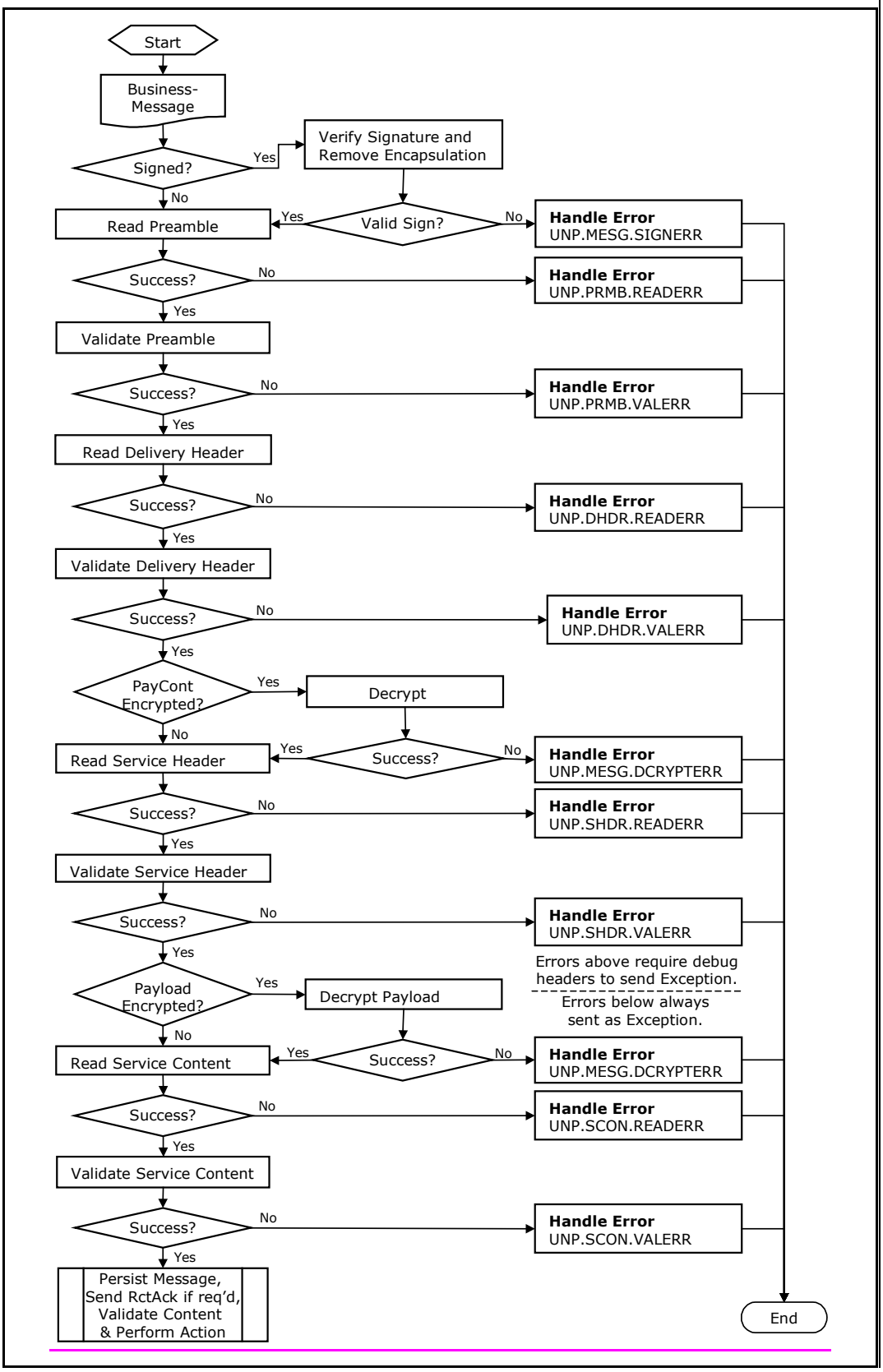


Figure 18. Entire Message Processing Flow

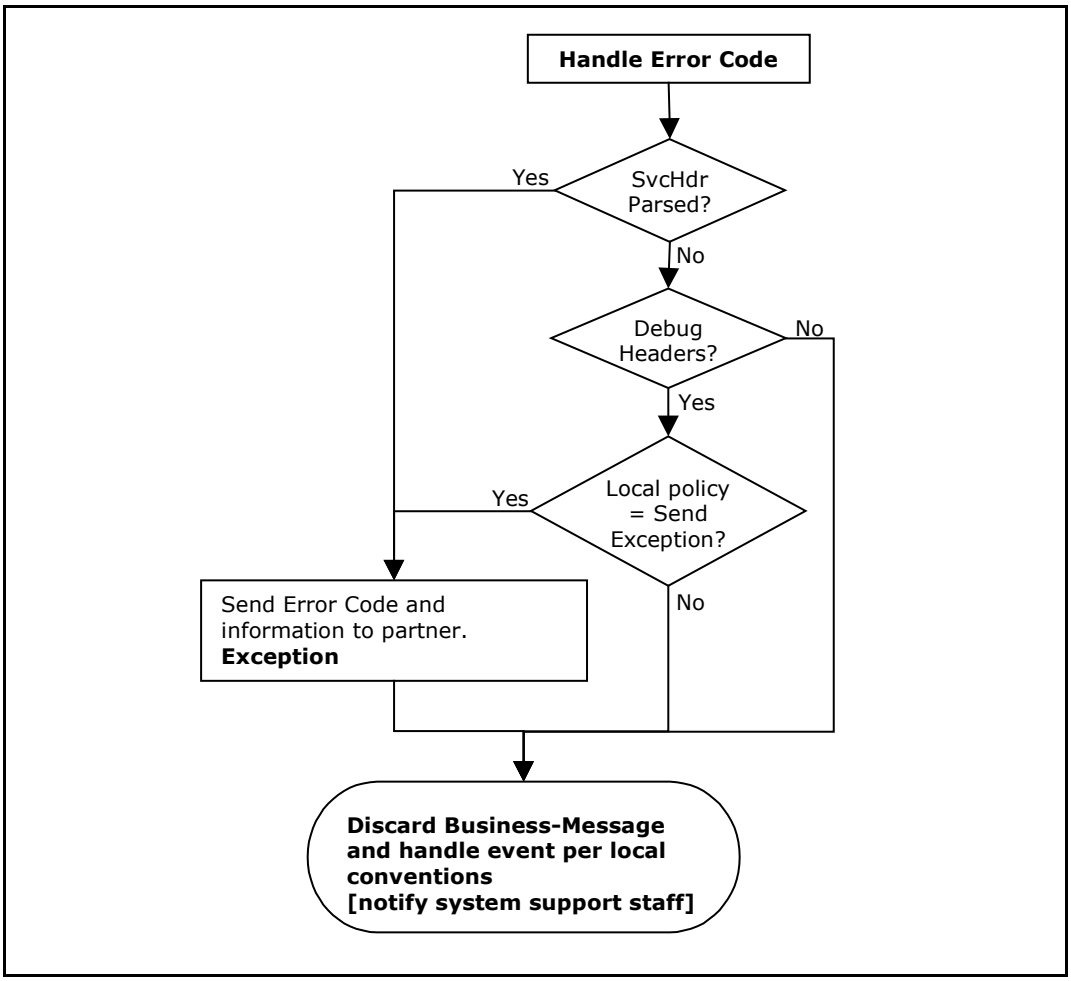


Figure 19. "Handle Error" Flow

2.3.5 Intermediary-Routed Business Messages

Intermediary-Routed messages are no different from the Peer-to-Peer messages. The intermediaries MUST always be able to read the Preamble and the Delivery Header. This is all the information needed to identify that this is a RosettaNet message and to identify the sender and the recipient of the message. In the event that the Preamble or the Delivery Header cannot be read by the intermediary, the intermediary may not be able to act on the message. The Delivery Header contains a ~~unique~~ tracking number for each message which, in combination with the sender's identification, can be used by ~~that~~ the intermediary, the sender, and the receiver ~~can use~~ to identify the message uniquely for tracking purposes~~track a message~~.

When the intermediary receives a message, it identifies the sender and the receiver. It MAY process the message per local policy and/or per the contract with the sender or the receiver. RosettaNet does not specify what the intermediary does internally or how it is done. Once the intermediary determines to send the message to the intended recipient, it merely sends the message to the recipient.

The intermediary SHALL send the incoming message as received from the sender and SHALL NOT reconstruct or repackage the message. This is necessary in order to be able to achieve non-repudiation of origin and content and non-repudiation of receipt.

2.4 RosettaNet Business Message Transfer

This section specifies transfer protocols for RosettaNet Business Message exchange, and specifies which are mandatory and which are optional. It also provides debug header specifications for use in certain situations.

One of the intentions of RNIF 2.0 is to de-couple the packaging (encoding) of the RosettaNet Business Message from the delivery or transfer of the RosettaNet Business Message. In doing so, more flexibility is provided to the implementers to select the transfer mechanism that best meets the requirements of a particular implementation. (See Appendix D for further rationale behind this approach.)

As a result, the concept of transfer independence is introduced. With transfer-independence, the RosettaNet Business Message defined in section 2.1 MUST be delivered to the receiving trading partner exactly as it was generated by the sender. To facilitate this, a transfer binding or envelope (transfer level header) specification, within which the transfer-independent RosettaNet Business Message MUST be transported end to end, and the transfer interface usage details are provided for each of the transfer protocols supported by RosettaNet. In the current release, RosettaNet specifies transfer binding and other details for HTTP and SMTP transfer protocols, with the intent to add support for more transfer protocols in the future. Use of additional transfer protocols is not considered RosettaNet-compliant until such time as these new protocol bindings are ~~specified~~published in a future RNIF release or addendum to this specification ~~and adopted through RosettaNet's development/approval/validation methodology~~.

In addition to allowing for maximum flexibility through transfer protocol independence, RNIF 2.0 also provides for maximum compatibility by specifying one transfer protocol that all solution providers MUST implement. Specifically, this protocol is HTTP. This guarantees that all RNIF 2.0-compliant trading partners can count on support for at least one transfer protocol (HTTP) being available from all solution providers.

This section also defines RNIF 2.0 debug-headers to be used at the transfer protocol level. These headers provide additional information on the content being transferred, to assist the implementers in the deployment effort. This document specifies details of these headers for HTTP and SMTP transfer protocols, as well as general guidelines for implementing debug headers for other transfer mechanisms.

2.4.1 Synchronous Response Messages

The RosettaNet PIP model is primarily based on an asynchronous message exchange mechanism, where reliable messaging is accomplished by means of separate acknowledgment message exchanges, as described in other parts of this specification. However, a need for transmitting the business response synchronously has already been identified by some of the more recent PIP specifications. Hence RNIF 2.0

specifies the transfer protocol level binding needed to perform synchronous exchange of messages also.

Of the two protocols (SMTP and HTTP) for which RNIF 2.0 specifies transfer protocol level bindings, HTTP is the only protocol that can support synchronous message exchanges. Hence the transfer bindings for synchronous message exchanges are specified and applicable to the HTTP transfer protocol only. It also follows that PIP implementers requiring synchronous message exchanges MUST use the HTTP transfer mechanism until RosettaNet specifies support in the future for other transfer protocols that would support synchronous message exchanges.

2.4.2 HTTP Transport Binding Specification

This section specifies the HTTP transfer envelope or the transfer-level headers to be used when transferring a RosettaNet message through HTTP transfer protocol. As noted earlier, all solution providers MUST support HTTP transfer protocol.

All trading partners MUST be able to use (i.e., exchange action and signal messages) this transfer protocol. Trading partners MAY use alternate protocols by agreement with selected trading partners (e.g., non-RNIF-compliant trading partners).

RNIF 2.0 RECOMMENDS the use of HTTP version 1.1 for the improvements it offers over HTTP version 1.0. Stable implementations of HTTP version 1.1 are widespread at this point. However, RosettaNet recognizes that many implementations still support HTTP version 1.0 only. Hence, use of HTTP version 1.0 is also permitted, and HTTP 1.1 implementations can downgrade the service to 1.0 level. However RosettaNet urges trading partners and solution providers to move to HTTP 1.1.

Note that the specifications and examples that follow show the use of the HTTP 1.1 specification. However, use of HTTP 1.0 version in all such places ~~should~~ MUST be considered valid as well. The HTTP protocol request lines (e.g., HTTP POST), as required by the HTTP 1.0 and later versions of the specification, MUST explicitly supply the version of the HTTP protocol. In addition, to facilitate cross-compatibility between HTTP versions, RosettaNet REQUIRES that every HTTP request contain a valid Content-Length header field.

2.4.2.1 Outbound HTTP Binding

When using the HTTP protocol, the outbound RosettaNet messages are transferred via an HTTP POST request to a trading-partner-specified URL.

The message to be transferred is transmitted as the “body” of the HTTP POST request. The following MIME headers are to be used with the HTTP POST request:

```
Content-Type: multipart/related;
              boundary=”any-value-appropriate”;
              type=”value”
x-RN-Version: RosettaNet/V02.00
x-RN-Response-Type: sync or async
Content-Length: nnnn
```

The MIME Content-Type MUST be multipart/related, with the two required parameters: “type” and “boundary”. The value for the type parameter MUST be the same as the MIME content-type for the RosettaNet message being transmitted. For RNIF 2.0, the only valid values are “multipart/related” and “multipart/signed”. The value for the boundary parameter MUST follow the standard MIME specification; note that RosettaNet does not specify the boundary parameter. However, care must be taken to use a value for the boundary that does not conflict with the potential boundary values in the RosettaNet message being transmitted. The x-~~RosettaNet~~RN-Version header with a value of “RosettaNet/V02.00” MUST be specified.

The Content-Length header, if used, MUST be in compliance with RFC 2616.

The x-RN-Response-Type header can take only one of the two values: “sync” or “async” (case insensitive). The x-RN-Response-Type header when present with the value “sync” specifies to the receiver of the message that the sender of the message requires a synchronous response. However, the x-RN-Response-Type header is OPTIONAL, and if not present, the value of the header defaults to “async” or the usual asynchronous message exchange mechanism. If the x-RN-Response-Type header is present with a value other than “sync” or “async” (case insensitive), the HTTP request should be rejected with a response code of 400 (Bad Request).

Other standard (HTTP-compliant) MIME headers MAY be used as mutually agreed by the trading partners. However, these headers are not significant from a RosettaNet message transfer/envelope perspective. Implementers are explicitly prohibited from attaching any significance that makes the RosettaNet message transport end-to-end dependent on those headers. See also debug-headers described in section 2.4.2.6~~2.4.2.6~~~~2.4.2.6~~2.4.2.6~~2.4.2.5~~ below.

Following the standard MIME convention, the MIME header and parameter names and values are not case-sensitive. The order in which the parameters occur is also not significant.

Note: In the following examples, all headers for the “body” RosettaNet message are **not** shown.

Example 12. HTTP Post of a RosettaNet Message

```
POST http://TPserver.TPcompany.com/cgi-bin/rosettanetservice HTTP/1.1
Content-Type: Multipart/related;
              boundary=”RN-HTTP-Boundary”; type=”multipart/related”
Content-Length: nnnn
x-RN-Version: RosettaNet/V02.00
x-RN-Response-Type:  async

--RN-HTTP-Boundary

[RosettaNet Business Message goes here]

--RN-HTTP-Boundary--
```

Example 13. HTTP Post of Unsigned RosettaNet Message

```
POST /servlet/RNInBoundServlet HTTP/1.1
Host: partnerA.name.com
Content-Type: Multipart/related;
```



```
                Boundary="RN-HTTP-Body-Boundary";
                type="multipart/related"
x-RN-Version: RosettaNet/V02.00
x-RN-Response-Type:  async
Content-Length: 1896

--RN-HTTP-Body-Boundary
Content-Type: multipart/related;
                boundary="RN-Outer-Boundary";
                type="application/xml"
Content-Description: This is the RosettaNet Business Message

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RN-Preamble-Header"
Content-ID: <value>

[Preamble Header instance goes here]

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RN-Delivery-Header"
Content-ID: <value>

[Delivery Header instance goes here]

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RN-Service-Header"
Content-ID: <value>

[Service Header instance goes here]

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RosettaNet-Service-Content"
Content-ID: <value>

[Service Content instance goes here]

--RN-Outer-Boundary
Content-Type: image/gif
Content-Transfer-Encoding: Base64
Content-ID: <value>

[Attachment goes here]

--RN-Outer-Boundary--

--RN-HTTP-Body-Boundary--
```

Example 14. HTTP Post of Signed RosettaNet Message

```
POST http://partnerB.name.com/servlet/RNInBoundServlet HTTP/1.1
Content-Type: multipart/related;
                type="multipart/signed";
                boundary="RN-HTTP-Boundary";
x-RN-Version: RosettaNet/V02.00
x-RN-Response-Type:  async
Content-Length: 18899

--RN-HTTP-Boundary
Content-Type: multipart/signed;
```

```
                boundary="RN-Signature-Boundary";
                protocol="application/pkcs7-signature";
                micalg=sha1
Content-Description: This is a Signed RosettaNet Business Message

--RN-Signature-Boundary
[The Business Message to be signed goes here]
[Business Message Payload Container + Preamble
[packed in MIME multipart/related construct]

--RN-Signature-Boundary
Content-Type: Application/pkcs7-signature; name="detached.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

[The base64-encoded PKCS7 Detached Signature]

--RN-Signature-Boundary--

--RN-HTTP-Boundary--
```

2.4.2.2 Processing Inbound HTTP Posts

The HTTP processor on the receiving side MUST verify the posted message for correct content-type and other MIME headers at the transfer level. The HTTP processor MUST also make sure the HTTP body matches the Content-Length (if specified). See section 2.4.2.6~~2.4.2.6~~~~2.4.2.6~~2.4.2.6~~2.4.2.5~~ below for details on dealing with debug headers.

If the HTTP headers are incorrect, or the content length specified does not match, or for any other errors related to receiving the HTTP posted message successfully, error codes as specified in the HTTP 1.1 RFC 2616 MUST be returned. Additionally RosettaNet RECOMMENDS that 1xx responses should never be returned, that 2xx responses SHOULD be limited to 200 and 202 (200 in the case of synchronous HTTP requests and 202 in the case of asynchronous, as further discussed below), and that 3xx, 4xx and 5xx error conditions must be dealt with in the usual way, governed by the local policy. See the description below for specific guidelines on HTTP errors to be returned. See section 2.6 for a detailed description of how to handle error conditions (exception handling).

If the value of the x-RN-Response-Type header is "async" (or if the x-RN-Response-Type header is not present in the HTTP POST), and the posted message is successfully received completely, without any errors, ~~the message MUST be persisted in a non-volatile medium and~~ an HTTP response code of "202 Accepted" MUST be returned. For asynchronous messaging, in the case of HTTP based transfer, acknowledgment and response messages are returned in separate HTTP POST requests. Hence, a "202 Accepted" is the correct HTTP status code to be returned.

~~If the value of the x-RN-Response-Type header is "sync", and the receiver does not support synchronous message exchanges, an HTTP error with the error code 501 (Not Supported) MUST be returned. Otherwise the receiver MUST attempt to process the request for a synchronous response. Please note that the request for synchronous response could be in error, as the support for synchronous responses must be explicitly called out in the PIP specification.~~

2.4.2.3 Processing Inbound Synchronous HTTP Posts

Following are additional guidelines for handling "sync" requests:

- If the value of the x-RN-Response-Type header is "sync", and the receiver does not support synchronous message exchanges, an HTTP error with the error code 501 (Not Supported) MUST be returned. Otherwise the receiver MUST attempt to process the request for a synchronous response. Please note that the request for synchronous response could be in error, as the support for synchronous responses must be explicitly called out in the PIP specification.

- If the value of the x-RN-Response-Type header is "sync", and the requested PIP does not support synchronous response mode, then an exception with error code UNP.MESG.RESPTYPERR MUST be returned.

- If the received message is processed successfully, the response (MIME-packaged RosettaNet Business Message) MUST be conveyed on the same HTTP connection with a 200 OK response code. See the example below for the format of the HTTP response.

- If the received message does not pass authentication or authorization checks, the receiver should either return an HTTP error with "403" response code or close the connection without a response, according to local policy.

- If the grammar/schema validation of the incoming message fails, an Exception (General Exception) signal (MIME-packaged) with an appropriate error code (as described in section 2.6) ~~should~~ MUST be returned with the HTTP response code 200 OK.

- If the business content validation step fails or an error occurs while processing (performing) the request, an Exception (General Exception) signal with an appropriate error code (as described in section 2.6) ~~should~~ MUST be returned with the HTTP response code 200 OK.

- For one-action PIPs, a Receipt Acknowledgment signal may be ~~sent~~ returned with an HTTP response code 200 OK, if called for in the PIP specification; a response code 200 OK with no entity-body should be sent otherwise as positive response. For two-action PIPs, only business response messages (no Receipt Acknowledgments) can be returned as a positive response with an HTTP response code 200 OK. For both one-action and two-action PIPs, an exception signal message MUST be sent with a response code 200 OK for conditions requiring to report exceptions, as described above.

- If an entity body is returned as an HTTP response, a Content-Length header field MUST be included. The x-RN-Version and x-RN-Response-Type header fields MAY appear in the response, but are not required.

Example 15. HTTP Synchronous Response

```
HTTP/1.1 200 OK
Content-Type: Multipart/related;
              boundary="RN-HTTP-Boundary"; type="type"
Content-Length: nnnn
```

```
--RN-HTTP-Boundary

[RosettaNet Response Business Message goes here]

--RN-HTTP-Boundary--
```

The x-RN-Version and x-RN-Response-Type headers MAY be omitted from the synchronous response, as shown.

Refer to section 2.6 for further details on handling synchronous requests and for guidelines on handling one-action and two-action PIPs in synchronous exchanges.

~~2.4.2.3~~2.4.2.4 HTTP Synchronous Exchanges & the Message Sender

The following are some guidelines for the message sender of HTTP based synchronous message exchanges:

- The sender should receive an HTTP response code other than 200 OK, for HTTP transfer related errors.
- The sender should expect to receive Exception Signal messages in addition to business response messages, with an HTTP 200 OK response code.
- For one-action PIPs the sender ~~may~~ MUST receive a Receipt Acknowledgment signal with a 200 OK if called for in the PIP specification, or a 200 OK with no body otherwise as positive response. For two-action PIPs, only business response messages (no Receipt Acknowledgments) can be returned as a positive response, returned with a 200 OK. For both one-action and two-action PIPs, the sender MUST expect to receive an exception signal message also (instead of a positive response) with a 200 OK.
- If the sender receives no response within the timeout constraints as specified in the PIP specification, or if the connection is dropped or times out, this could be due to a failure or error condition at the receiver. If the sender believes that it is a valid request, the sender must close the HTTP session if not already terminated and the request ~~may~~ MAY be sent again, as a new instance of the subject PIP.

Refer to section 2.6 for further details.

~~2.4.2.4~~2.4.2.5 Transfer-Level Security

If additional transfer-specific security is desired, Secure Sockets Layer (SSL) protocol v.3 or any backward-compatible successors (such as TLS v.1.0) MAY be used. A minimum of SSL v.3 MUST be made available by solution providers.

~~2.4.2.5~~2.4.2.6 Debug Header as an Extension-Header in HTTP

The Debug-header provides additional information to the recipient of the RosettaNet Business Message via HTTP headers.

The RosettaNet Exception signals are asynchronous. That is, if an action message is sent by Partner A to Partner B, errors in the message are indicated by Partner B to Partner A asynchronously. These exceptions can only be sent by the recipient of the original message if (minimally) the Service Header of the incoming message could be read successfully. Errors that occur before successful reading of the Service Header would result in the sender timing out waiting for the Receipt Acknowledgment. As a value-add, solution providers MAY choose to provide a feature that enables the recipient of an action message to notify the sender if there is a problem unpacking a message. This can be done by the sender setting the RosettaNet debug header as an HTTP extension header. The recipient MAY then use this information to send an exception to the sender (if there was an error while unpackaging the message) even if the service header was not read completely.

The debug header is intended to be used during initial setup and testing, so that the trading partner receiving a message can send an exception to the trading partner who sent the message even if the service header was not successfully read.

However, if this feature is not made available in a solution, the solution will not be deemed non-compliant. Similarly a receiving trading partner MAY not wish to use this feature. This is also acceptable. For security reasons, debug headers SHOULD NOT be used in production mode. Debug headers if received during production mode SHOULD be ignored.

Debug headers MUST NOT be set while sending signals, in order to avoid an infinite loop.

The following is the form of the extension header ~~and the corresponding parameters to be used for the purpose of debugging.~~:

```
x-RN-Debug-Mode: Yes; ~~x-RN-PIP-Code~~<parameter>~~=~~<value~~>; x-RN-PIP-Instance-ID = <value>; x-RN-Activity-Code = <value>; x-RN-Action-Code = <value>; x-RN-Action-Instance-ID = <value>; x-RN-Partner-ID = <DUNS>;~~>~~; <parameter>=<value>~~;~~ ...
```

The parameters of the debug header are shown in Table , along with the XPATH-style locations from which the corresponding values are to be taken.

Table 5. Debug Header Parameters

Parameter Name	Value (Element Location)
x-RN-PIP-Code	/ServiceHeader/ProcessControl/pipCode/GlobalProcessIndicatorCode
x-RN-PIP-Version	/ServiceHeader/ProcessControl/pipVersion/VersionIdentifier
x-RN-PIP-Instance-ID	/ServiceHeader/ProcessControl/pipInstanceId/InstanceIdentifier
x-RN-Message-Tracking-ID	/DeliveryHeader/messageTrackingID/InstanceIdentifier
x-RN-Activity-Code	/ServiceHeader/ProcessControl/ActivityControl/BusinessActivityIdentifier
x-RN-Action-Code	/ServiceHeader/ProcessControl/ActivityControl/MessageControl/Manifest/ServiceContentControl/ActionIdentity/GlobalBusinessActionCode

Parameter Name	Value (Element Location)
x-RN-Sending-Partner-ID	/DeliveryHeader/messageSenderIdentification/PartnerIdentification/GlobalBusinessIdentifier
x-RN-Sending-Partner-Location-ID	/DeliveryHeader/messageSenderIdentification/PartnerIdentification/locationID/~~FreeFormText~~Value
x-RN-Initiating-Partner-ID	/ServiceHeader/ProcessControl/KnownInitiatingPartner/PartnerIdentification/GlobalBusinessIdentifier
x-RN-Initiating-Partner-Location-ID	/ServiceHeader/ProcessControl/KnownInitiatingPartner/PartnerIdentification/locationID/~~FreeFormText~~Value
x-RN-PIP-Payload-Binding-ID	/ServiceHeader/ProcessControl/partnerDefinedPIPPayloadBindingId/ProprietaryReferenceIdentifier

Refer to section ~~2.1.3.3~~2.1.3 for element descriptions. Each parameter MUST be present in the debug header, and its value exactly duplicated from the given location, if the respective Service Header or Delivery Header element is present. Parameters whose corresponding elements are not present MUST NOT appear in the debug header.

~~Note that the above~~The debug header MUST follow~~s~~ standard MIME header conventions, paying particular attention to quoting and long-line folding. The order in which the parameters are listed is of no significance (they MAY appear in any order), and parameter values are to be treated as case-sensitive.

~~2.4.2.6~~2.4.2.7 Compliance Summary

This summary is for convenience only and is not guaranteed to contain all compliance statements. For complete compliance knowledge, read the entire specification.

HTTP POST method with the MIME content-type of "multipart/related" MUST be used to transmit RosettaNet messages over HTTP. The "type" and boundary parameters MUST be specified with multipart/related content-type. The type parameter MUST match the MIME content-type of the RosettaNet Business Message being transmitted. The only valid values are "multipart/related" and "multipart/signed".

Content-Length MIME header MAY be specified with the HTTP POST and if specified, MUST match the length of the body posted. See RFC 2616 for details on use of Content-Length with HTTP POST.

Other standard (HTTP-compliant) MIME headers MAY be used as needed by the trading partners based on mutual agreements. However, these headers are not significant from a RosettaNet message transfer perspective. Solution providers are explicitly prohibited from attaching any significance to these additional headers that makes the RosettaNet message transport end-to-end dependent on those headers

All solution providers MUST provide support for HTTP transport. HTTP 1.1 or HTTP 1.0 level support is required.

Support for secure transport for HTTP is also mandatory for solution providers. SSL v.3 or any backward-compatible successor (such as TLS v.1.0) MUST be supported.

All solution partners must be able to correctly read and interpret the HTTP header response type (x-RN-Response-Type). In the event the solution does not support synchronous responses, it ~~must~~ MUST be able to return the HTTP status code 501 (Not Implemented).

2.4.3 SMTP Transport Binding Specification

This section specifies the SMTP transfer envelope or the transfer-level headers to be used when transferring a RosettaNet message through SMTP. RosettaNet messages are transmitted over SMTP by building an RFC 822-compliant email (SMTP/MIME) message, with the SMTP Transport Headers specified below forming RFC 822 message envelope/"headers" and the RosettaNet message to be transported forming the "body" of the RFC 822 message. Any RFC 822-compliant headers other than the ones specified in the section below MAY be used as needed. However, all the headers specified in the section MUST be used.

While use of SMTP is widespread, a number of SMTP implementations can still only support 7-bit data transmissions. Hence care MUST be taken to content-transfer-encode the binary and 8-bit content portions of RosettaNet messages if they will be transferred using SMTP.

2.4.3.1 SMTP Transport Envelope

The following MIME/RFC-822 headers MUST minimally be used to encapsulate the RosettaNet message for transmission via SMTP.

```
MIME-Version: 1.0
Content-Type: multipart/related;
              type="value";
              boundary="any-value-appropriate"
x-RN-Version: RosettaNet/V02.00
Content-Length: nnnn
From: value
To: value
```

Additional MIME/RFC-822 headers MAY be used as needed by the trading partners based on mutual agreement. However, these headers are not significant from a RosettaNet message transfer/envelop perspective. Solution providers are explicitly prohibited from attaching any significance to additional headers that make the RosettaNet message transfer end-to-end dependent on those headers.

Partners MUST agree on and exchange the email addresses to be used when sending RosettaNet messages over SMTP transport. The sending partner's email address MUST be specified in the "From" header field and receiving partner's address MUST be specified in the "To" header field. The recipient should be aware that SMTP headers (including the From header field) are susceptible to spoofing.

A content-type value of "multipart/related" MUST be used, with the two required parameters "type" and "boundary". The value for the type parameter MUST be the

same as the MIME content-type for the RosettaNet message being transmitted. For RNIF 2.0, the only valid values are “multipart/related” and “multipart/signed”. The “boundary” parameter is also needed by the MIME multipart/related content-type and MUST be specified. The value for the boundary parameter MUST follow the standard MIME specification and is not specified by RosettaNet. However, care MUST be taken to use a value for the boundary that does not conflict with the potential boundary values in the RosettaNet message being transmitted. The x-~~RosettaNet~~RN-Version header with a value “RosettaNet/V02.00” MUST be specified.

Following the standard MIME convention, the MIME header and parameter names and values are not case-sensitive. The order in which the parameters occur is also not significant

The entire RosettaNet message MUST be added as the body of an RFC-822 compliant email message with headers specified above (adding any OPTIONAL headers as needed). The message so built is sent over SMTP to a partner-specified SMTP server (as agreed in advance by the trading partners), to be delivered to the email address specified in the “To” header field.

Note: In the following examples, all headers for the “body” RosettaNet message are **not** shown.

~~Example 15.~~Example 16. RosettaNet Message Encased in SMTP Envelope

```
MIME-Verson: 1.0
From: sendingpartner@sendcompany.com
To: receivingpartner@receivingcompany.com
Content-Type: Multipart/related;
              boundary=”2934792834”;
              type=”body-MIME-type”
x-~~RosettaNet~~RN-Version: RosettaNet/V02.00
Content-Length: 5609

--2934792834

[The RosettaNet-Message to be sent goes here]

--2934792834--
```

~~Example 16.~~Example 17. Unsigned RosettaNet Message in SMTP Envelope

```
MIME-Verson: 1.0
From: sendingpartner@sendcompany.com
To: receivingpartner@receivingcompany.com
Content-Type: Multipart/related;
              Type=”multipart/related”;
              Boundary=”RN-SMTP-Body-Boundary”
x-~~RosettaNet~~RN-Version: RosettaNet/V02.00
Content-Length: 1896

--RN-SMTP-Body-Boundary
Content-Type: multipart/related;
              boundary=”RN-Outer-Boundary”;
              type=”application/xml”
Content-Description: This is the RosettaNet Business Message

--RN-Outer-Boundary
Content-Type: Application/XML
```



```
Content-Location: "RN-Preamble"
Content-Description: This is the XML that is the Preamble

[Preamble goes here]

--RN-Outer-Boundary
Content-Type: Application/XML
Content-Location: "RN-Delivery-Header"
Content-ID: <value>

[Delivery Header instance goes here]

--RN-Outer-Boundary

Content-Type: Application/XML
Content-Location: "RN-Service-Header"
Content-ID: <value>

[Service Header instance goes here
--RN-Outer-Boundary

[Service Content goes here]

--RN-Outer-Boundary--

--RN-SMTP-Body-Boundary--
```

~~Example 17.~~Example 18. Signed RosettaNet Message in SMTP Envelope

```
MIME-Verson: 1.0
From: sendingpartner@sendcompany.com
To: receivingpartner@receivingcompany.com
Content-Type: Multipart/related;
              Type="multipart/signed";
              Boundary="RN-SMTP-Boundary"
x-~~RosettaNet~~RN-Version: RosettaNet/V02.00
Content-Length: 18899

--RN-SMTP-Boundary
Content-Type: multipart/signed;
              boundary="RN-Signature-Boundary";
              protocol="application/pkcs7-signature";
              micalg=sha1
Content-Description: This is a Signed RosettaNet Business Message

--RN-Signature-Boundary
[The Business Message to be signed goes here]
[Business Message Payload Container + Preamble
 packed in MIME multipart/related construct]

--RN-Signature-Boundary
Content-Type: Application/pkcs7-signature; name="detached.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

[The base64-encoded PKCS7 Detached Signature]

--RN-Signature-Boundary--

--RN-SMTP-Boundary--
```

2.4.3.2 Transfer-Level Security

SMTP does not naturally support transfer-level security. Hence trading partners are encouraged to explore the possibility of using HTTP with SSL. However, encrypting the content to be transferred would provide some level of privacy if SMTP needs to be used. Refer to section 2.3 for details on how to encrypt and decrypt the content.

2.4.3.3 Transfer-Level Error Handling

Email-based message transfer is a store-forward-based message delivery mechanism and the SMTP messages need not be sent directly between the source and the eventual recipient's SMTP nodes (due to SMTP routing involved). Hence, trading partners cannot rely on any synchronous transport level errors (analogous to HTTP response/error codes) being returned. Therefore, trading partners MUST have a mechanism in place to handle undeliverable email messages sent to each other. Delivered messages with content problems SHOULD, however, result in the recipient sending separate RosettaNet Exception business signals. If desired, trading partners could use the SMTP Delivery Status Notification (DSN) mechanism (see RFC 1891) to request that the recipient notify the sender of SMTP message delivery status. Partners could also use the SMTP Message Disposition Notification (MDN) mechanism as needed. These are part of the standard SMTP message delivery mechanism / standard and can be used by trading partners as needed and feasible, based on their SMTP set-ups. RosettaNet does not provide any explicit specification in this respect.

Many SMTP servers have a delivery timeout of several days, which may be longer than the performance controls specified in a PIP. Also, messages sent via SMTP might not be delivered in the order in which they are sent. Trading partners should take these constraints into consideration prior to choosing SMTP as the delivery method.

2.4.3.4 Debug Header as an Extension-Header in SMTP

The Debug Header provides additional information to the recipient of the RosettaNet Business Message via SMTP headers.

Refer to the HTTP debug header section (section 2.4.2.6~~2.4.2.6~~~~2.4.2.6~~2.4.2.6~~2.4.2.5~~) for the specification~~rationale of the use~~ and usage of this header.

~~The Debug headers MUST NOT be set while sending signals in order to avoid an infinite loop.~~

~~The following is the extension header and the corresponding parameters to be used for the purpose of debugging. Note that these elements are always present in a Service Header.~~

~~x-RN-Debug-Mode: Yes; x-RN-PIP-Code = <value>; x-RN-PIP-Instance-ID = <value>; x-RN-Activity-Code = <value>; x-RN-Action-Code = <value>; x-RN-Action-Instance-ID = <value>; x-RN-Partner-ID = <DUNS>;~~

~~Note that the above header follows standard MIME header conventions.~~

2.4.3.5 Compliance Summary

This summary is for convenience only and is not guaranteed to contain all compliance statements. For complete compliance knowledge, read the entire specification.

The MIME content-type of "multipart/related" MUST be used to transmit RosettaNet messages over SMTP. The "type" and boundary parameters MUST be specified with the multipart/related content-type. The type parameter MUST match the MIME content-type of the RosettaNet Business Message being transmitted. The only valid values are "multipart/related" and "multipart/signed".

x-RN-Version header with a value of "RosettaNet/V02.00" MUST be specified

Content-Length MIME header MUST be specified and MUST match the length of the RosettaNet message included in the body.

MIME-Version header with a value of "1.0" MUST be specified.

All MIME header and parameter names are case-insensitive. The order of parameters in a header is insignificant. Both are per standard MIME conventions.

Other standard MIME/RFC-822 compliant headers MAY be used as needed by the trading partners based on mutual agreements. However, these headers are not significant from a RosettaNet message transfer/envelop perspective. Solution providers are explicitly prohibited from attaching any significance to these additional headers that makes RosettaNet message transfer end-to-end dependent on those headers.

2.4.4 Transfer Protocol Independence and Other Transfer Mechanisms

The transfer protocol-independent nature of the RosettaNet Business Message enables addition of support for other transfer protocols as they are adopted by RosettaNet in the future, without impacting the format of the message.

To facilitate the use of other private transfer mechanisms (e.g., file-based) by which trading partners may exchange messages between themselves (without direct support from solution providers), solution providers SHOULD make transfer protocol-independent RosettaNet Business Messages available for delivery by other transfer mechanisms. The means by which the messages are made available to the alternate transfer mechanisms is not specified by RosettaNet. Similarly, solution providers SHOULD provide hooks to process the messages received via other transfer mechanisms. Again the means by which this is done is not specified by RosettaNet.

2.4.5 General Guideline for Debug Mode for Other Transport Protocols

The purpose of the debug header is to supply critical information from the Service Header in case the Service Header for an incoming message could not be read successfully. This information MUST minimally include: PIP Code, PIP Instance ID, Activity Code, Action Instance ID, Instance ID, Sending Partner ID.

2.5 Business Signal Specifications & Process Control PIPs

This section identifies and specifies current business signals as well as PIPs that are used in controlling the process of PIP business exchanges.

In the execution of PIPs that carry out specific business functions (e.g., 3A4 -- Manage Purchase Order), it might be necessary for certain system-level acknowledgment messages or exception messages to be returned to one or both parties to the PIP. These are referred to as "business signals" and are distinct from the business action messages that are defined by each business PIP.

Additionally, there are other classes of RosettaNet PIPs that do not perform any business-related function (e.g., a possible class of "maintenance" PIPs which could cover such RosettaNet functions as dictionary maintenance). One of these classes is called "process control PIPs"; these PIPs perform various system-level administrative tasks that aid in the execution of business PIPs.

These business signals and process control PIPs are part of the implementation framework and are more fully described in this section. The Message Guidelines, DTDs, and (in the case of Process Control PIPs) PIP specifications are published in separate files and available through the normal PIP access channels.

For additional insight into how these business signals and process control PIPs are used, see section 2.6.

2.5.1 Business Signals

Business signals are positive and negative acknowledgment messages that are sent in response to business actions. There is one positive business signal (Receipt Acknowledgment) and one negative business signal (Exception); all other RosettaNet messages are business actions. In contrast to business actions, all business signals are RosettaNet-specified and carry no content from other sources.

Whether the Receipt Acknowledgment signal is required for a given Business Action is specified in the corresponding PIP specification. Detailed specifications on when a specific kind of signal should be sent are provided in section 2.6; additionally, further description of the uses of these signal is available in ~~the~~ section 2.3.

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

2.5.1.1 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 the 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 (it is almost always required).

To send a Receipt Acknowledgment for the appropriate business action, use the PIP, action, and activity information in the received message's Service Header.

See the following documents for the Receipt Acknowledgment specifications.

- Acknowledgment of Receipt DTD (AcknowledgmentOfReceipt_MS_~~B~~V02_00.dtd)
- Acknowledgment of Receipt Message Guideline (AcknowledgmentOfReceipt_MG_~~B~~V02_00_00.htm)

DOCUMENT TYPE DEFINITION

```
<!ENTITY % common-attributes "id  CDATA #IMPLIED" >
<!ELEMENT ReceiptAcknowledgment ( NonRepudiationInformation? ) >
<!ATTLIST ReceiptAcknowledgment xmlns CDATA #FIXED
          "http://www.rosettanet.org/RNIF/V02.00~~/~~" >
<!ELEMENT NonRepudiationInformation ( OriginalMessageDigest ) >
<!ELEMENT OriginalMessageDigest ( #PCDATA ) >
```

TREE STRUCTURE FROM MESSAGE GUIDELINE

```
1  1     ReceiptAcknowledgment
2  0..1  |-- NonRepudiationInformation
3  1     | |-- OriginalMessageDigest
```

See the actual Message Guidelines for descriptions of these elements.

2.5.1.2 Exception

See the following documents for the Exception specifications.

- Exception DTD (Exception_MS_~~B~~V02_00.dtd)
- Exception Message Guideline (Exception_MG_~~B~~V02_00_00.htm)

DOCUMENT TYPE DEFINITION

```
<!ENTITY % common-attributes "id  CDATA #IMPLIED" >
<!ELEMENT Exception (
          ExceptionDescription ,
          GlobalExceptionTypeCode ) >
<!ATTLIST Exception xmlns CDATA #FIXED
          "http://www.rosettanet.org/RNIF/V02.00~~/~~" >
<!ELEMENT ExceptionDescription (
          errorClassification ,
          errorDescription ,
          offendingMessageComponent? ) >
<!ELEMENT errorClassification ( GlobalMessageExceptionCode ) >
<!ELEMENT GlobalMessageExceptionCode ( #PCDATA ) >
<!ELEMENT errorDescription ( FreeFormText ) >
<!ELEMENT FreeFormText ( #PCDATA ) >
<!ATTLIST FreeFormText xml:lang CDATA #IMPLIED >
<!ELEMENT offendingMessageComponent ( GlobalMessageComponentCode ) >
<!ELEMENT GlobalMessageComponentCode ( #PCDATA ) >
```

```
<!ELEMENT GlobalExceptionTypeCode ( #PCDATA ) >
```

TREE STRUCTURE FROM MESSAGE GUIDELINE

```
1  1     Exception
2  1     |-- ExceptionDescription
3  1     | |-- errorClassification.GlobalMessageExceptionCode
4  1     | |-- errorDescription.FreeFormText
5  0..1  | |-- offendingMessageComponent.GlobalMessageComponentCode
6  1     |-- GlobalExceptionTypeCode
```

See the actual Message Guidelines for descriptions of these elements.

2.5.2 Process Control PIPs

Process Control PIPs are designed to be sent by either party to a PIP dialogue to notify the other party of events that affect the execution of the business PIP or to ascertain status of a business PIP that is believed to be in process.

These PIPs follow the business PIP naming conventions, and belong to the cluster "0" and the segment "A".

As of this writing, only one such PIP exists.

2.5.2.1 0A1: Notification of Failure (NoF)

See the following documents for the PIP 0A1: Notification of Failure specification~~s~~.

- PIP Spec (0A1_Spec_~~B~~V02_00_00.doc)
- DTD (0A1_MS_~~B~~V02_00_FailureNotification.dtd)
- Message Guideline (0A1_MG_~~B~~V02_00_00_FailureNotification.htm)

2.6 Flow of RosettaNet Business Messages

RosettaNet Partner Interface Processes (PIPs) are implemented by the exchange of business messages in specific sequences and specific timeframes. These business messages contain both control and content information to meet PIP requirements.

A PIP defines one or more business activities involving two or more partner roles. A business activity consists of one or more business actions executed in the sequence specified by RosettaNet.

RosettaNet PIPs follow a specific choreography of action and signal message exchange. A PIP instance begins by a partner starting the first action in an activity in the PIP and continues until all the actions in the activity are completed successfully or an action fails.

An action execution results in a business action message being sent from one trading partner (Trading Partner A) to another (Trading Partner B) and if specified by the PIP,

an acknowledgment signal being sent from the recipient (Trading Partner B) of the original message to its sender (Trading Partner A), to indicate the fact that the action message has been validated from a security point of view and that all of the base-level validation rules described in section 2.1 have been applied successfully. Depending on the PIP, the recipient of the original message (Trading Partner B) may have to perform a response action. The response action message would then be sent from Trading Partner B to Trading Partner A, with Trading Partner A possibly acknowledging receipt of the response action message. This completes the entire PIP instance. This entire action message and signal exchange constitutes the choreography of the PIP.

It is important to note that the overlapped execution of multiple instances of the same PIP or related PIPs between two trading partners is not addressed here. PIP specifications state the semantics for executing multiple instances of the same PIP or related PIPs in overlapping timeframes, and should provide real-world examples of such concurrent execution where appropriate. If such semantics are not met, then the situation should be treated as an action performance failure.

2.6.1 Asynchronous Single-Action (Simplest) Activity

The simplest choreography is an asynchronous single-action PIP activity. That is, an activity where one action message is sent from Partner A to Partner B and the Receipt Acknowledgment is sent from Partner B to Partner A. When this complete set of one action message and one signal message have been exchanged successfully between these trading partners, the PIP instance is deemed complete at both ends. One commonly used single-action PIP is PIP2A1: Distribute New Product Information.

While the Receipt Acknowledgment indicates successful receipt and grammar/schema validation of an action message by an action message recipient, the exception message indicates an error in processing of the action message. An exception sent by Partner B in the above scenario indicates failure of the above PIP instance at both the partners' systems.

To be exact, in the above scenario the PIP reaches completion state at Partner A upon receipt of the Receipt Acknowledgment. Partner B reaches completion only after finishing its internal processing of the action message. These two events could happen at different times. If Partner B has returned a Receipt Acknowledgment to Partner A and then encounters an error in its internal processing of the action message, then it MUST initiate an instance of the Notification of Failure PIP to inform Partner A of the error. This is because Partner A will have completed its PIP instance once it receives the Receipt Acknowledgment.

If a PIP does not specify Receipt Acknowledgment (such as in an information distribution scenario), then the action and the PIP are complete for the sender once it has successfully transmitted the message, and is complete for the receiver as soon as the action message is received and processed by the receiver. The receiver MUST initiate an instance of the Notification of Failure PIP if it encounters any error while processing the action message.

2.6.2 Asynchronous Two-Action Activity

A slightly more complex scenario would be the above case, but with the addition of Trading Partner B sending a response action to Trading Partner A. That is, Trading Partner A initiates the activity by sending Action Message X to Trading Partner B. Trading Partner B sends a Receipt Acknowledgment to Trading Partner A and then later sends a response action message Y to Trading Partner A. Trading Partner A then sends a Receipt Acknowledgment signal to Trading Partner B. In this case, Trading Partner A completes execution of the PIP instance immediately after processing message Y. (If Trading Partner A fails to send a Receipt Acknowledgment signal to Trading Partner B, Trading Partner A will still close the PIP after processing message Y.) Trading Partner B completes execution of the PIP instance after receiving the Receipt Acknowledgment for message Y. (If Trading Partner B does not receive a Receipt Acknowledgment signal from Trading Partner A, Trading Partner B will continue resending message Y until a Receipt Acknowledgment is received or until Trading Partner B decides to issue a Notification of Failure.) One commonly used two-action PIP is PIP3A4: Manage Purchase Order.

If Trading Partner B has returned a Receipt Acknowledgment to Trading Partner A and then encounters an error in its internal processing of action message X, it MUST send an exception to Trading Partner A. Since Trading Partner A is still waiting for the response action, it is unnecessary for Trading Partner B to initiate an instance of the Notification of Failure PIP. On the other hand, once Trading Partner A has returned a Receipt Acknowledgment confirming receipt of the response action to Trading Partner B, then any subsequent error encountered by Trading Partner A in processing the response message MUST trigger an instance of the Notification of Failure PIP.

2.6.3 Synchronous One-Action/Two-Action Activity

By default, PIP interactions between two trading partners are asynchronous. When HTTP is used as transport, each business action message or business signal message flows over a separate HTTP connection. In order to support PIPs such as PIP 2A9 (Query EC Technical Information) that require immediate responses and optimized use of network bandwidth, RNIF 2.0 allows for synchronous PIPs over HTTP transport. An initiator of a synchronous single-action PIP MUST specify that the message exchange is to be completed synchronously in the HTTP header (x-RN-ResponseType: sync). Similarly, an initiator of a synchronous two-action PIP that does not require Receipt Acknowledgment MUST specify in the HTTP header that the response be returned synchronously ~~in the HTTP header~~. If the responder of a synchronous activity does not support synchronous interaction at all, it MUST return the HTTP status code “501 Not Implemented”.

Each PIP specifies whether the entire exchange of messages is synchronous or asynchronous. In the absence of any definition in the PIP specification, the default SHALL be that all exchanges are asynchronous. The following rules apply to the processing of the “x-RN-Response-Type” HTTP header:

1. When the PIP specification requires a response to be asynchronous, an initiating partner SHALL always designate the interaction to be asynchronous and SHALL not designate synchronous.

2. When the PIP specification requires a Receipt Acknowledgment to be synchronous but does not require a substantive response, an initiating partner SHALL always designate the interaction to be synchronous and SHALL not designate asynchronous.

3. When the PIP specification requires a response to be synchronous and no Receipt Acknowledgment at all, an initiating partner SHALL always designate the interaction to be synchronous and SHALL not designate asynchronous.

4. When the PIP specification allows responses to be either asynchronous or synchronous, an initiating partner MAY designate either asynchronous or synchronous. The receiving partner SHALL provide the response in the manner indicated.

In general, it is possible to execute a PIP instance synchronously only if the PIP has a single action, or if the PIP has two-actions and neither Receipt Acknowledgment nor Nnon-Rrepudiation of Receipt is required. This does not preclude the use of digital signatures in synchronous response mode for purposes of Non-Repudiation of Origin and Content (either single- or two-action PIPs), or for Non-Repudiation of Receipt (single-action PIPs only).

Retries, even if specified by a PIP, are not allowed for in a synchronous interaction. Timeouts must result in the initiator closing the connection and terminating the PIP. If retries are performed, they MUST be PIP-level retries, with each being a new PIP instance.

2.6.4 Handling Failures

Failures can occur at any point in PIP execution, as discussed in the following subsections. Two methods of handling failure are provided in RNIF 2.0: sending an exception signal or initiating a Notification of Failure (NoF) PIP.

To determine whether an exception signal should be sent or whether to initiate a NoF, the following guidance may be useful. In general, send an exception signal if it is the case that the trading partner should still be executing the PIP in question; initiate a NoF if it is possible that the other trading partner is not executing the PIP (e.g., has not yet begun processing or has completed processing already).

2.6.4.1 Retries and Timeouts

Note: The following discussions on retries and timeouts apply only to asynchronous PIP activities. Synchronous PIPs SHOULD specify a Time To Perform that does not inherently exceed the reasonable/acceptable time for leaving a HTTP connection open, and a Retry Count of 0. Time to Acknowledge, if specified, should be identical to Time to Perform.

In order to achieve transfer independence, transfer protocol-specific acknowledgments are not attached any receipt semantics. That is, if a HTTP acknowledgment is received, it only means successful “delivery” of the message and nothing more. Hence, the sender is dependent on a Receipt Acknowledgment to infer that the

message has been received, read and validated (for grammar and schema) successfully by the recipient. The Receipt Acknowledgment happens asynchronously (i.e., uses a different "connection" from the incoming message). Therefore, the sender has to "wait" for the Receipt Acknowledgment. Under certain conditions, when the recipient is not able to decipher anything about the incoming message, the recipient cannot inform the sender about the error (as the sender or the PIP context may not be identifiable). Under such conditions, the sender cannot wait indefinitely for the acknowledgment. This necessitates the concept of "timeout".

Timeouts can occur when the sender does not receive the Receipt Acknowledgment after a particular time. The reason could be either that the recipient never received the original message (this could be difficult to infer depending on the transfer protocol used); the recipient was not able to "read" the message; the recipient had a problem sending the Receipt Acknowledgment; or the Receipt Acknowledgment never reached the sender of the original message.

Nevertheless, in order to ensure a more reliable message delivery, the sender MUST retry sending the message until either the Receipt Acknowledgment or an Exception is received, or until all allowable retries are exhausted~~, subject to the additional constraint that the Time to Perform has not expired~~.

Once the initiator of a two-action PIP receives the Receipt Acknowledgment for the initial action, it should wait for a response action message from the responder. A~~gain,~~ Time to Perform timeout can occur while waiting for the response action message.

When a timeout occurs and retries are no longer allowable, the PIP instance on the sender's side ends in a failure state. In order to ensure that the other partner does not continue with the process, the sender MUST initiate an instance of the Notification of Failure PIP.

In general, retries and timeouts for a PIP are governed by the Time To Acknowledge, Time To Perform, and Retry Count parameters in the PIP's Business Activity Performance Controls table:

- Time to Acknowledge[1] refers to the time duration within which a partner role that initiates a role interaction MUST receive acknowledgment that a Business Document is received by a responding partner role. This time is measured from the instant the action message has been sent successfully. That is, once an action message has been delivered successfully, the sender expects to receive a Receipt Acknowledgment before Time to Acknowledge has elapsed. In a one-action PIP, only the initiator needs to monitor the Time to Acknowledge. In a two-action PIP, ~~the initiator and the responder each send out an action message.~~ Time to Acknowledge applies individually ~~both~~ to both the initiator and the responder; each ~~with respect to~~ wait~~sing~~ for a Receipt Acknowledgment in reply to their respective initiating and responding ~~for the sent~~ action messages.

~~□ Time to Acknowledge refers to the time within which a partner role that initiates a role interaction MUST receive acknowledgement that a Business Document is~~

[1] This is a slight misnomer. A more appropriate term would have been Time to Receive Receipt Acknowledgment

~~received by a responding partner role. This time is measured from the instant the action message has been sent successfully. That is, once an action message has been delivered successfully, the sender expects to receive a Receipt Acknowledgement before Time to Acknowledge has elapsed.~~

- Time to Perform refers to the time duration within which the PIP activity MUST be successfully performed. It is measured from the Message Date Time value found in the Delivery Header within the action message that initiates the PIP instance. For a two-action PIP, Time to Perform is interpreted as the time for receipt of the response action message by the initiator. The initiator should consider the PIP instance as failed if no response action is received before Time to Perform elapses. Only the initiator of a PIP instance is required to ensure that the PIP instance is completed within the allowable Time to Perform. The responder of a PIP instance SHOULD NOT test an incoming action message that initiates a PIP instance for expiration, nor ~~does it have to~~must it abort execution of the PIP instance ~~should~~if the private process does not respond before Time to Perform expires. Nevertheless, it is assumed that clocks at the initiator and at the responder are synchronized closely enough ~~in "sync"~~ that the~~y can trust each other's~~ Delivery Header Date Timestamps can be relied upon. The method for such synchronization is outside the scope of this specification. ~~The timestamp must be generated before signature calculation but as close to the time of first attempted transport by the RosettaNet implementation as possible.~~ If specified for a single-action PIP, Time to Perform MUST be ignored, as only the Time to Acknowledge and Retry Count parameters are relevant to this type of exchange.

~~Time To Perform refers to time within which the PIP activity MUST be successfully performed. It is measured from to the Message Date Time value found in the Service Header within the action message that initiates the PIP instance. For a two-action PIP that expects an action message from the responder, the initiator should consider the PIP instance as failed if no response action is received before Time to Perform elapses. Only the initiator of a PIP instance is required to ensure that the PIP instance is completed within the allowable Time to Perform. The responder of a PIP instance SHOULD NOT test an incoming action message that initiates a PIP instance for expiration, nor does it have to abort execution of the PIP instance should the private process not respond before Time to Perform expires.~~

- Retry Count refers to the number of times an action message MAY be retransmitted (in addition to the initial attempt) due to timeout waiting for Receipt Acknowledgment. Thus, if the Retry Count is 3, an action message may be sent a total of 4 times. For a two-action PIP, the Retry Count applies both to the initiator for sending the request action message and to the responder for sending the response action message. This interpretation is different from RNIF 1.1 where retries for two-action PIPs may be triggered both by timeout waiting for Receipt Acknowledgment (action-level retries) and by timeout waiting for response action (activity-level retries), and the Retry Count is used to govern both action-level and activity-level retries. RNIF 2.0 eliminates activity-level retries because they are ~~deemed difficult to implement correctly. There are too many implementation situations that make rollback of previous attempts of the same PIP instance and exception handling extremely difficult for back-end systems. This can result in undesirable duplicate processing as well as other "out-of-synch"~~

~~conditions~~unnecessary; a Receipt Acknowledgment is sufficient indication that a message has been persisted and restarting the activity would be pointless. Furthermore, handling of activity-level retries in the context of more complex activities would be even more problematic than it would be with today's simpler one- and two-action patterns.

~~Retry Count refers to the number of times an action message MAY be retransmitted (in addition to the initial attempt) due to timeout waiting for Receipt Acknowledgement. Thus, if the Retry Count is 3, an action message may be sent a total of 4 times.~~

An RNIF 2.0 implementation MAY also perform transport (i.e., HTTP/SMTP) level retries if non-fatal transport level status codes result from transmission attempts. However, the frequency of such retries is outside the scope of the RNIF 2.0 specification and SHOULD be addressed via trading partner agreements.~~-~~ If communication failure persists after all agreed-upon transport-level retries have been exhausted, then an instance of the Notification of Failure PIP MUST be executed if the failed message is an action message or an exception message. If the failed message is a Receipt Acknowledgment, Notification of Failure is not executed, because the intended recipient is expected to time out and possibly resend the action message.

2.6.4.2 Other Failure Conditions and Notification of Failure

An RNIF implementation SHOULD maintain sufficient state information related to open PIP instances so that on recovery from a system failure, the progress of open PIP instances can be resumed. For example, if an initiator was waiting for Receipt Acknowledgment for a PIP instance and was supposed to retry sending the action message at time *t*, then on restart it SHOULD continue to wait for Receipt Acknowledgment until time *t* before retrying. Of course, if time *t* has already passed, then retransmission of the action message SHOULD happen immediately, provided that the Time to Perform has not yet expired and that the allowable retries have not been exhausted.

In the asynchronous single-action scenario, Partner A “completes” or “deems complete” its PIP instance a little ahead of Partner B. This leaves the possibility that Partner B could encounter a failure while processing the action message (i.e., after Partner A has attained the PIP completion stage). Though the probability of this may be low, it is nevertheless something that the PIP choreography should provide for. Since the PIP is deemed completed at Partner A, Partner A will no longer expect to receive signals from the same PIP instance. Hence, Partner B MUST initiate an instance of the Notification of Failure PIP to Partner A in such cases.

Similarly, in the asynchronous two-action scenario, Partner A could encounter an error while processing Message Y (i.e., after sending Receipt Acknowledgment for message Y to Partner B). Since at this point Partner B would have completed its PIP instance, Partner A MUST initiate an instance of the Notification of Failure PIP to indicate the failure in the PIP instance.

Notification of Failure is only intended as an out-of-band mechanism to signal error conditions. It MUST NOT be used when a responding party encounters an exception while processing a business document request. An exception is be used in such cases.

It is RECOMMENDED that the communications channel, application server or network, (or combinations thereof) used by the Notification of Failure PIP instance be different from ~~the one~~those used for regular PIP instances. This is to enable reporting of failures caused by communication problems. Trading partners MAY also agree that still another mechanism be used to report inability to execute Notification of Failure PIPs. ~~In an e-business environment, this alternate communications channel SHOULD at least be interpreted to mean communicating with an application server that is different from the application server that has not serviced the original business document request~~. Trading partners SHOULD, however, agree on the exact nature of these~~is~~ "alternate communication~~s~~ channels". In addition, they SHOULD specify the legal meaning and the private process logic for Notification of Failure triggered by the execution of individual PIP types.

In order to avoid an "infinite loop" scenario, another instance of Notification of Failure MUST NOT be initiated in response to an error encountered during the execution of an instance of Notification of Failure.

2.6.5 Receipt Acknowledgment

The Receipt Acknowledgment confirms that the grammar and schema rules applicable to the message received are satisfied. See base level validation described in section 2.1. Trading partners MAY optionally agree to validate other constraints specified in the message guidelines that are beyond the scope of base-level validation prior to sending Receipt Acknowledgments.

Nevertheless, the above acknowledgment is the result of verification of a static set of syntactical and data validation rules. It does not confirm any semantic validation of the message as such validation can vary from trading partner to trading partner and can depend heavily on the end system. This type of validation is called "content validation" or validation of the content of a business action message against the organization's internal business rules. If an action message does not pass content validation, and this is not the last action within the PIP, then the recipient MAY return an exception to the sender. The exception type is "General Exception" and the error code to use is PRF.DICT.VALERR~~UNP.SCON.VALERR~~. If this is the last action within the PIP, then the recipient MUST initiate a Notification of Failure because the sender ~~has~~ may have already received the Receipt Acknowledgment and ~~has~~ closed its PIP instance.

The fact that the Receipt Acknowledgment was sent by the receiver of the business message is good enough to infer that the business message was indeed delivered and "read" successfully by the intended recipient. Also, RNIF 2.0 requires that the recipient of a business action message save a persistent copy of the business action message after grammar and schema validation, so as to avoid unnecessary retries once the PIP initiator has received the Receipt Acknowledgment for the initial business action message. Thus, the Retry Count specified for a PIP is interpreted as the number of action-level retries only. The initiator of a PIP ~~MAY~~ MUST re-send an action message if no Receipt Acknowledgment is received within the Time to Acknowledge, subject to the Retry Count and Time to Perform constraints. Likewise, the responder of a two-action PIP ~~MAY~~ MUST re-send a response action message under the same conditions. Unlike RNIF 1.1, RNIF 2.0 does not provide for retrying at the activity-

level. ~~Consequently, there is no longer any notion of an attempt count in the Service Header for an action message either.~~

2.6.6 Handling Retries and Late Acknowledgments

As established earlier, the trading partner sending an action message retries the message until either a Signal (Receipt Acknowledgment or Exception) is received or a timeout condition occurs. Hence, the receiver MUST be prepared to receive the same action message more than once. In such a case, if the action requires a Receipt Acknowledgment, the Receipt Acknowledgment (or Exception if there is a failure) MUST be resent.

Also, as mentioned earlier, the PIP choreography is independent of the transfer or transport mechanisms. Therefore, it is possible that for a given request, the Receipt Acknowledgment can arrive after the response message. This MUST NOT be deemed as an out-of-order message. If the response is received before the Receipt Acknowledgment and the request action requires non-repudiation of receipt, then any of the following suggested approaches MAY be followed.

A response that arrives before the Receipt Acknowledgment MAY either be queued for processing until the Receipt Acknowledgment is received or processed immediately. If the response is processed immediately, then the process SHALL NOT be completed until the Receipt Acknowledgment is received, since the Receipt Acknowledgment contains the digest information for non-repudiation of receipt. These approaches are suggestive only and the implementer is free to choose a similar approach as long as the result is the same (i.e., the response SHALL NOT be rejected unless a timeout occurs waiting for the Receipt Acknowledgment).

2.6.7 Receipt Acknowledgment and General Exception Error Codes

Table 5 shows the mandatory Error Codes and their associated descriptions for Receipt Acknowledgment exceptions and general exceptions.

~~Table 5.~~Table 6. Exception Error Codes

Error Code (Case Insensitive)	Description
PKG.MESG.GENERR	Error during packaging – General error
PRF.ACTN.GENERR	Error during action performance – General Error
PRF.DICT.VALERR	Error during action performance – Validating the Service Content against a PIP-specified dictionary
UNP.MESG.GENERR	Error during unpackaging – General error
UNP.MESG.SIGNERR	Error during unpackaging – Verifying the signature of the RosettaNet Business Message
UNP.PRMB.READERR	Error during unpackaging – Reading the Preamble
UNP.PRMB.VALERR	Error during unpackaging – Validating the Preamble
UNP.DHDR.READERR	Error during unpackaging – Reading the Delivery Header

Error Code (Case Insensitive)	Description
UNP.DHDR.VALERR	Error during unpackaging – Validating the Delivery Header
UNP.SHDR.READERR	Error during unpackaging – Reading the Service Header
UNP.SHDR.VALERR	Error during unpackaging – Validating the Service Header
UNP.SHDR.MNFSTERR	Error during unpackaging – Verifying Manifest against the actual attachment body parts
UNP.MESG.SEQERR	Error during unpackaging – Validating the message sequence
UNP.MESG.RESPTYPERR	Unexpected Response type in the HTTP header
UNP.MESG.DCRYPTERR	Error Decrypting the message
UNP.SCON.READERR	Error during unpackaging – Reading the Service Content
UNP.SCON.VALERR	Error during unpackaging – Validating the Service Content

2.6.8 Interaction Diagrams

The diagrams in this section are intended to illustrate the general flow for both single-action and two-action activities in both asynchronous and synchronous interactions.

The FSV section of the PIP specification documents contain specific interaction diagrams detailing the normal flow of business messages (action and signal) between services performing the PIP partner roles. Those diagrams show which business action messages and business signal messages are part of the choreography of the PIP.

2.6.8.1 Asynchronous Interactions

Figure 20 and Figure 21 illustrate the high-level choreography of an asynchronous single-action activity and an asynchronous two-action activity, respectively. The boxes with solid boundaries represent steps that are executed in the public process space; those with dashed boundaries represent steps that are executed in the private process space.

It should be noted that the Responder side "Validate Message Structure" step in these figures actually encompasses all of the validation steps shown in Figure 19. It includes all base-level validation specified in section 2.1, plus optional schema-level validation (if any) that may have been agreed between the two trading partners. Validation using business rules, however, is responsibility of the private process and is assumed to be included in the "Process Action Message" step.

The "Handle Error" step in Figure 20 and Figure 21 roughly corresponds to the "Handle Error" Flow in Figure 19.

The steps labeled "Requesting Business Action Message", "Receipt Acknowledgment", "Responding Business Action Message" in Figure 20 and Figure

21 represent transfer of the packaged business action/signal message over HTTP or SMTP. An error may occur in sending any of these messages. Depending on the trading partner agreement, transport-level retries MAY be used. If the communication error persists ~~and this happens~~ while attempting to send~~ing~~ an action message ~~or an exception message~~, then an instance of the Notification of Failure PIP SHOULD be executed to notify the trading partner of the communication failure. If the message is a ~~Receipt Acknowledgment~~business signal, the sender ~~is not supposed to~~ SHOULD NOT initiate Notification of Failure. Instead, the intended recipient is expected to time out and ~~possibly~~ retry, if appropriate, before finally initiating the Notification of Failure ~~sending the action message~~. ~~If the error occurs while sending the PIP's first action message, then the error SHOULD be handled internally. An instance of Notification of Failure MAY also be initiated to notify the trading partner of the communication failure. If the message is a Receipt Acknowledgement, and if the error does not get resolved in a number of retries, the sender is not supposed to initiate Notification of Failure. Instead, the sender SHOULD just give up sending the message, thereby causing the intended recipient to time out and possibly retry sending the action message.~~ If this is the last Receipt Acknowledgment within a two-action PIP, the message initiator MAY close the PIP instance without reporting any error, even though the responder has not received the Receipt Acknowledgment.[2] The responder in this case SHOULD retry sending the response action message and might eventually have to initiate Notification of Failure. However, Notification of Failure in this case does not necessarily mean that the business transaction has to be aborted. It only signals that there is a problem with completing the PIP's choreography. If the initiator has already received the response action from the responder's initial or retried attempts, there is no reason to nullify the business transaction. For example, if the PIP specification calls for non-repudiation of receipt for the response action, then the responder's execution of the Notification of Failure might simply require the initiator to return the missing Receipt Acknowledgment (including the signed digest for the received action message) to the responder. This will have to be done through some other out-of-band mechanism not currently defined by RNIF 2.0.

It should be noted that the logic for dealing with communication failures described above is not captured in the figures in this section. ~~If the PIP specification calls for non-repudiation of receipt for the response action, then the responder's execution of the Notification of Failure might require the initiator to return the missing Receipt Acknowledgement (including the signed digest for the received action message) to the responder through some other out-of-band mechanism not currently defined by RNIF 2.0. The logic described above is not captured in the figures in this section.~~

Similarly, the exception-handling rule for business signals is such that when there are errors validating the Preamble, Delivery Header, or Service Header for grammar,

[2] Even if the transmission of the Receipt Acknowledgment is successful, the initiator cannot know positively when the responder has successfully validated the Receipt Acknowledgment for a certain time because the responder is not allowed to send an acknowledgment for the Receipt Acknowledgment. This "finish" state can only be inferred if the responder has not triggered NoF within (Retry Count + 1)~~4~~ times the PIP's Time to Acknowledge. Since the NoF may itself fail due to a variety of problems before human intervention takes place, another (NoF Retry Count + 1)~~4~~ times the NoF Time to Acknowledge may elapse before the initiator learns of the original failure and the subsequent NoF failure through an external channel.

content or sequence, the erroneous signal MAY be logged internally and essentially ignored. No exception SHOULD be sent to the sender of these signals, nor SHOULD the Notification of Failure PIP be initiated. The logic for ignoring incorrect signal messages again is not explicitly represented in any of the figures.

Likewise, logic necessary for implementing non-repudiation of origin and content / non-repudiation of receipt, checkpoint and restart, etc., are omitted from these figures to limit their complexity.

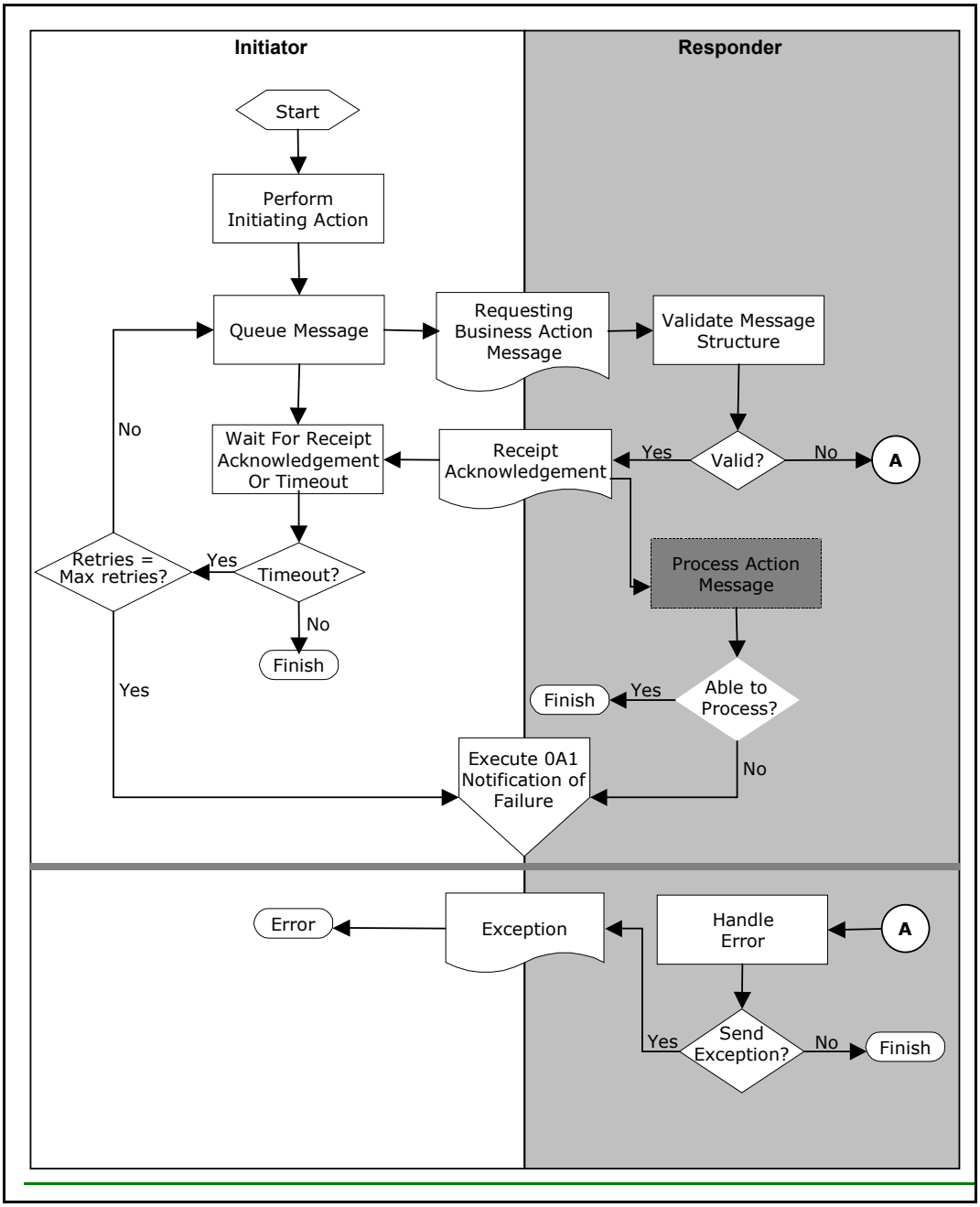


Figure 20. Single-Action Activity (Asynchronous)

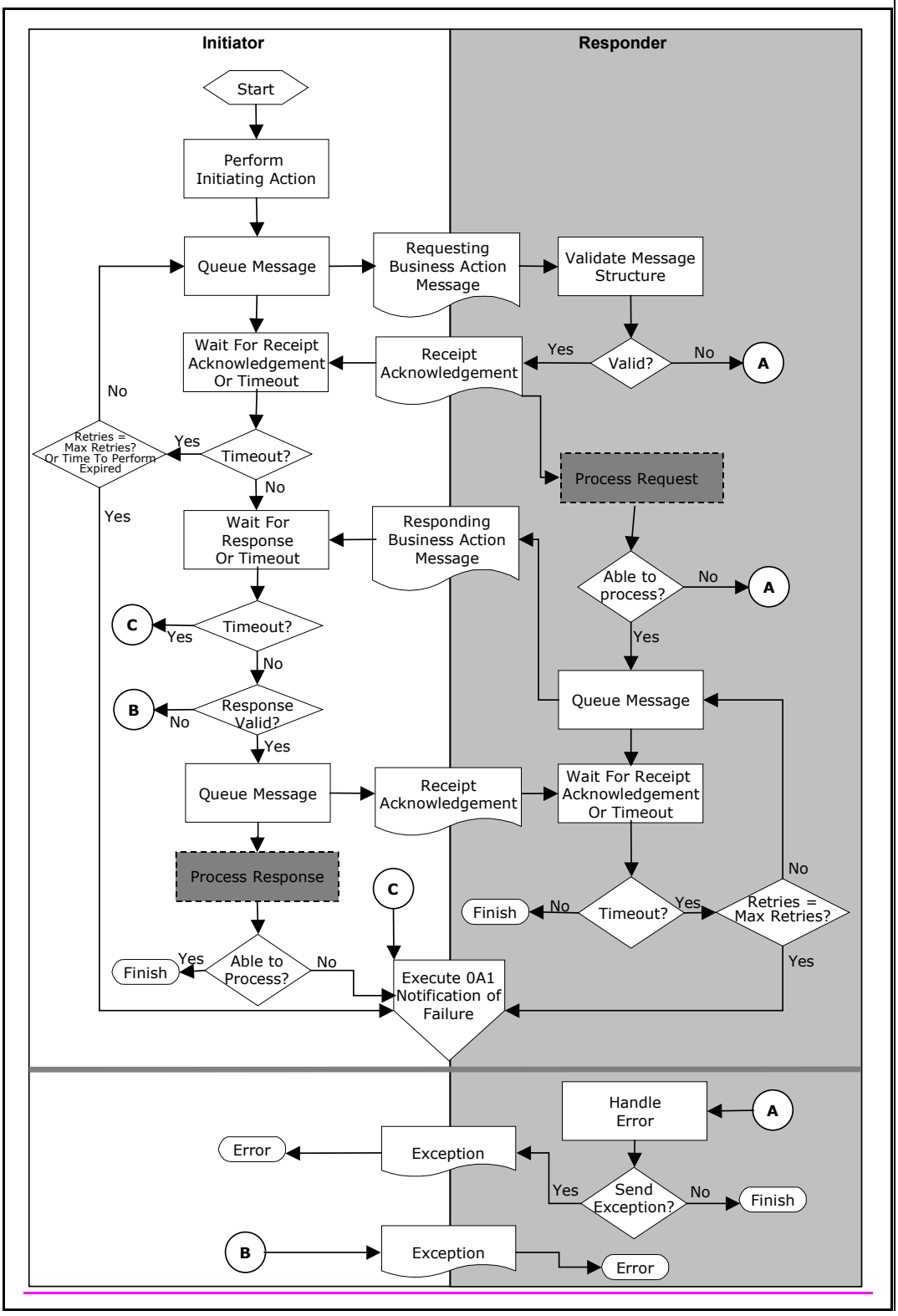


Figure 21. Two-Action Activity (Asynchronous)

2.6.8.2 Synchronous Interactions

Figure 22 and Figure 23 illustrate the high-level choreography of a synchronous single-action activity and a synchronous two-action activity, respectively. As before, the boxes with solid boundaries represent steps that are executed in the public process space while those with dashed boundaries represent steps that are executed in the private process space.

Since PIP interactions are by default asynchronous, an action message that initiates a PIP instance must explicitly specify in the HTTP header "x-RN-Response-Type" a value of "sync", if the PIP is single-action and the Receipt Acknowledgment (if any) is expected to be returned synchronously, or if the PIP is two-action and the Response action is expected to be returned synchronously.

If the HTTP header specifies synchronous response and this is disallowed in the PIP specification, then an exception along with a status code of 200 will be returned over the same HTTP connection. If the incoming message contains a signature and the signature cannot be verified, or if HTTPS is used as the transport and errors occur during handshaking, the connection is simply closed and no error code is returned. Otherwise, a single "403 Forbidden" HTTP response code is returned for signature verification and authorization errors. If the responder does not support synchronous interactions at all, a single "501 Not Implemented" HTTP response code is returned.

For a synchronous single-action PIP that requires Receipt Acknowledgment, detailed processing of the action message happens after the HTTP connection has been closed. Thus, if the processing is unsuccessful, an instance of the Notification of Failure PIP MUST be executed. A synchronous two-action PIP, on the other hand, completes processing of the incoming action message and can either return a Response action or an exception over the original HTTP connection. Therefore, there is no necessity for the responder to initiate the Notification of Failure PIP to report errors back to the initiator.

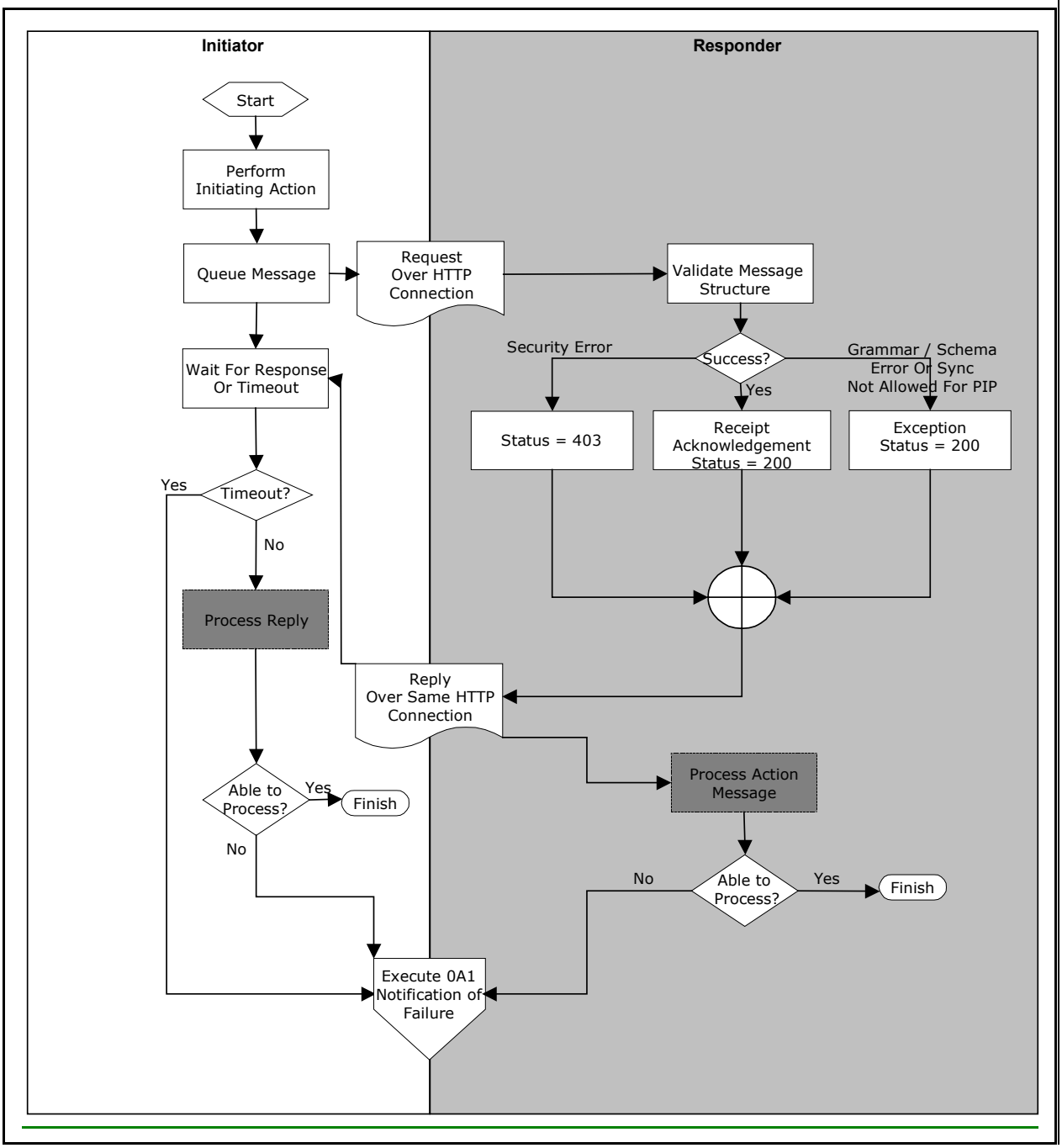


Figure 22. Single-Action Activity (Synchronous)

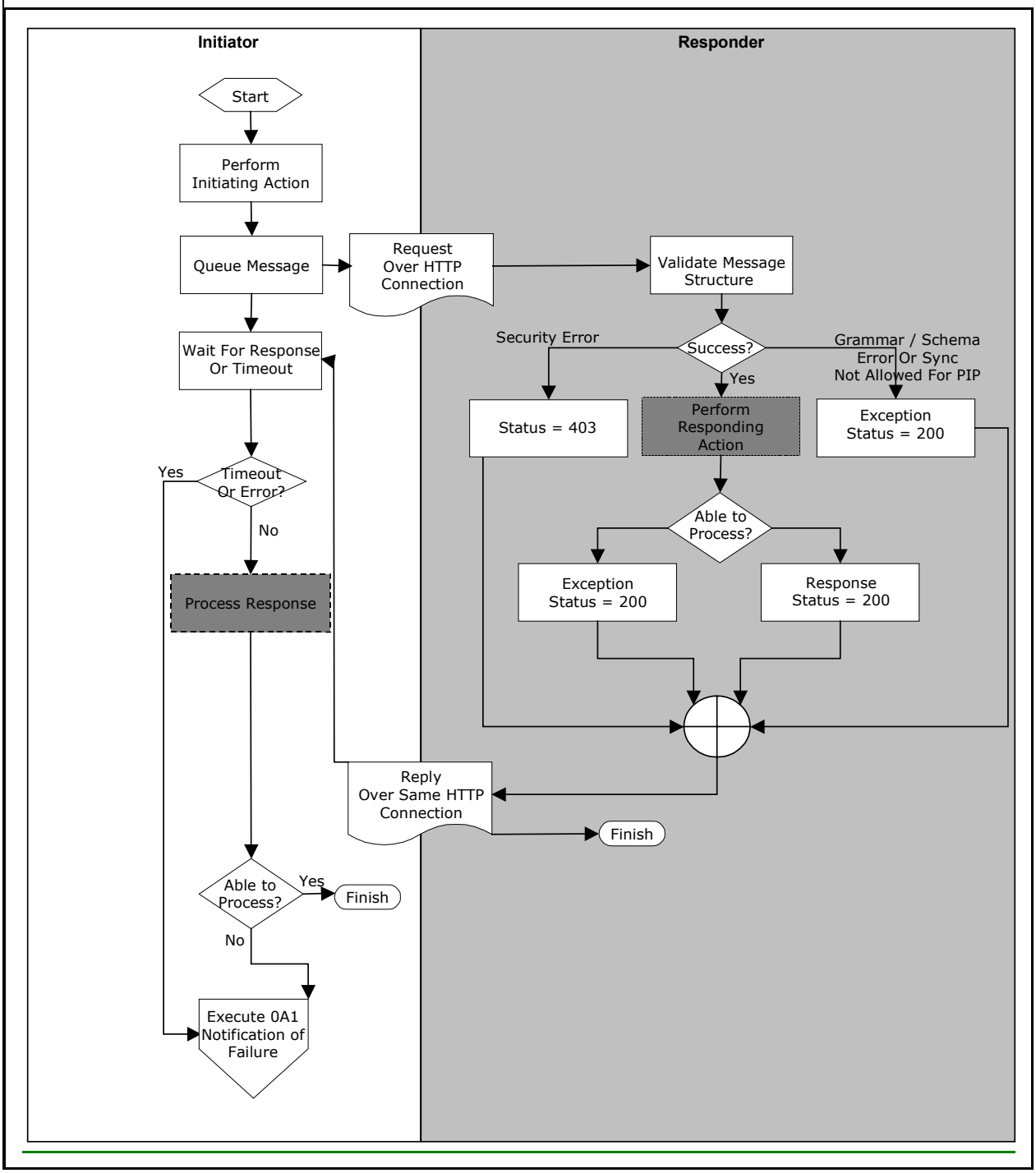


Figure 23. Two-Action Activity (Synchronous)

2.6.8.3 Notification of Failure Scenarios

To compensate for the lack of low-level details in ~~figures 20 to 23~~Figure 20 to Figure 23, ~~Table 6~~Table 7 provides a complete list of the scenarios under which Notification of Failure will be initiated for each corresponding type of PIP activity.

Table 7. Notification of Failure Scenarios

Type of Activity	Initiator	Responder
One Action (async)	1. Initiator fails in establishing communication with Responder for sending the action message. 2. Initiator times out waiting for Receipt Acknowledgment after exhausting retries for sending the action message.	1. Responder successfully sends Receipt Acknowledgment for action message, but thereafter fails in its further processing.
Two Actions (async)	1. Initiator fails in establishing communication with Responder for sending the action message. 2. Initiator times out waiting for Receipt Acknowledgment after exhausting retries for sending the action message.[3] 3. Initiator does not receive the response action message from Responder before Time to Perform expires.[4] 4. Initiator fails in processing the response action message after sending Receipt Acknowledgment to the Responder.	1. Responder fails in establishing communication with Initiator for sending the response action message. ~~Responder fails in establishing communication with Initiator for sending an exception signal message.~~ 2. Responder times out waiting for Receipt Acknowledgment after exhausting retries for sending the response action message.
One Action (sync)	1. Initiator fails in establishing communication with Responder for sending the action message. 2. Initiator does not receive the response action message beore Time To Perform expires.	1. Responder successfully sends Receipt Acknowledgment for the action message, but thereafter fails in its further processing.

[3] Example: With a Retry Count of 3 and Time to Acknowledge ~~set at~~of 2 hours, NoF may be triggered after (3 + 1) * 2 = 8 hours due to non receipt of Receipt Acknowledgment by the Initiator (assuming that are no communication failures that result in Notification of Failure). Since the sending of an action message is not instantaneous and may require transport-level retries, the NoF triggered by the Receipt Acknowledgment timeout may actually happen after more than 8 hours due to the way Time to Acknowledge is defined.

[4] Example: With a Retry Count of 3 and Time to Perform ~~set at~~of 24 hours, NoF may be triggered after 24 hours due to non receipt of the response action message from the Responder. This is different from the (3 + 1) * 24 = 96 hours for RNIF 1.1, which allows activity-level retries.

Type of Activity	Initiator	Responder
Two Actions (sync)	1. Initiator fails in establishing communication with Responder for sending the action message. 2. Initiator does not receive the response action message from Responder before Time To Perform expires. 3. Initiator fails in processing the response action message after ~~sending Receipt Acknowledgment to~~receiving the response from the Responder.	No scenario for Notification of Failure.

APPENDIX A KEY DIFFERENCES BETWEEN RNIF 1.1 & RNIF 2.0

This appendix outlines features that are either new in RNIF 2.0 or that have been substantially changed from RNIF 1.1.

Feature	1.1 Treatment	2.0 Treatment
Multiple Transfer Protocols	HTTP was the only transfer protocol supported.	New in 2.0. While all RNIF-compliant implementations must support HTTP, RNIF 2.0 provides guidelines for the use of other transfer protocols. SMTP specification is added in 2.0. Others would be added in future releases of RNIF.
Attachments	No explicit support. Private agreements needed to use.	Formal support added in 2.0 RNIF 2.0 provides for formal framework for attaching supporting documents to the business content (service content). These could be .pdf file, word document, or files in GIF TIF and other formats. RNIF 2.0 also defines a mechanism by which attachments could be referenced from the business content (XML documents).
Encryption of Service Content and Service Header	Not available.	New in 2.0. RNIF 2.0 recommends use of S/MIME based content enveloping scheme for encrypting the Service Content and also the Service Header as needed by the partners.
Support for Hubs and Delivery Header	Not available	New in 2.0. RNIF 2.0 adds a new header called Delivery Header and makes associated recommendations for use by partners when RosettaNet messages are sent through Hubs between trading partners.
Third-Party Service Content	Not available	New in 2.0. RNIF 2.0 adds support for shipping non-RosettaNet Service Content (e.g. business documents whose format is standardized by standard bodies other than RosettaNet), as sanctioned by RosettaNet, in RosettaNet PIPs.
Synchronous Transactions	Not available	New in 2.0. RNIF 2.0 permits synchronous exchange of request and response messages in a single HTTP session, if permitted by the PIP.

Digital Signature Packaging	Uses RosettaNet Object (RNO) format for signing / to attach detached (PKCS7) signatures to the RosettaNet business messages.	Uses standard S/MIME format for signing or attaching the signatures to the RosettaNet business messages.
Message Manifest	Not Available	New in 2.0. RNIF 2.0 adds support for Message Manifest that describes the payload contents.
Service Header		Service header is restructured in 2.0 to eliminate inconsistencies in 1.1 version and to add support for new features such as third-party content, attachments, message manifest.
Signals and Signal Fields		RNIF 2.0 eliminated the Acceptance Acknowledgment Signal. RNIF 2.0 also integrated all the Exception Signals into one schema (DTD) and Guideline specification and added a field to identify the specific signal being sent. RNIF 2.0 removes some of the RosettaNet action message-specific fields from signals to provide support for third-party service content. RNIF 2.0 adds an error code field to exception signals that can be used to return specific error condition codes with the signal.
Quality Of Service	Not Available	RNIF 2.0 adds a Quality of Service element to the Service Header as a placeholder hook for specifying dynamically negotiable Quality of Service parameters for the message exchange between trading partners. This is a placeholder at this point (for future backward compatibility), to be specified fully in a future version of the RNIF specification.
Retry Level	Activity-level retries.	RNIF 2.0 eliminated Activity-level retries and calls for individual Action level retries only. See section 2.6.
Exception Handling	Described in a Technical Advisory issued separately.	RNIF 2.0 integrates the description of Exception Handling and message flow into the specification. See section 2.6.
Debug Headers	Not Available	New in 2.0. RNIF 2.0 adds support for transfer-level debug headers that can be used during initial set-up by trading partners.

APPENDIX B REQUIRED PIP METAMODEL CHANGES

This version of the RNIF core specification introduces new functionality beyond RNIF 1.1. In order to allow newly designed PIPs to fully take advantage of the new features and to remove certain perceived inconsistencies, the PIP Metamodel has to be enhanced. This appendix identifies the changes that are expected to be applied to the existing PIP Metamodel.

B.1 Machine-readable PIP Specifications

Many of the elements in the Service Header of business action and signal messages make use of fields that are to be extracted from PIP specifications. It is assumed that in the near future PIPs will be published in machine-sensible XML formats so that the construction of business action and signal messages can be automated. As a pre-requisite, the grammar and schema to which all XML-based PIP specifications MUST conform will have to be specified.

B.2 Retry

In RNIF 1.1, retries are applied at the activity level. The Retry Count found in the Business Activity Performance Controls table of a PIP's Business Operation View determines the number of times a PIP activity can be retried due to timeouts waiting for Receipt Acknowledgments or Response Action messages. In other words, retries can happen both at the activity level and at the action level. In RNIF 2.0, only action-level retries happen as a result of timeouts waiting for Receipt Acknowledgments. The requirement that a recipient make a persistent copy of an action message before acknowledging is designed to eliminate expensive activity-level retries that may require the re-computation of digital signatures. Since retry is no longer an activity-level concept, its specification should be moved to the Message Exchange Controls table of each PIP's Functional Service View.

B.3 Encryption

RNIF 2.0 allows for the encryption of message payloads above the transport level. The Message Exchange Controls section of a PIP's Functional Service View therefore should include an attribute "Is Encryption Required?"

B.4 Synchronous versus Asynchronous

By default, the exchange of action and signal messages between business partners is asynchronous. However, activities with only one or two actions can optionally be completed over a single synchronous HTTP connection. The Message Exchange Controls section of a PIP's Functional Service View therefore SHOULD include an attribute "Use Synchronous Connection?" If this is not specified in the PIP, then it must be considered to be an asynchronous response. It should be noted that a

synchronous two-action activity MUST NOT require Receipt Acknowledgment. This implies that synchronous two-action activities do not support non-repudiation of receipt.

B.5 Acceptance Acknowledgment

RNIF 2.0 no longer supports the use of the Acceptance Acknowledgment concept for non-substantive acknowledgments of initial business actions. The Time to Acknowledge Acceptance attribute in the Business Activity Performance Controls table in the Business Operation View and the Time to Acknowledge Acceptance Signal in the Functional Service View therefore should be omitted for newly designed PIPs.

B.6 Non-Repudiation of Receipt

In the Business Activity Performance Controls table of a PIP's Business Operation View, the Acknowledgment of Receipt column should indicate whether Non-Repudiation is required for the initial action message or for all action messages within the PIP. Currently, there are some PIPs that specify Non-Repudiation for the request action message but not for the response action message in the Message Exchange Controls table in the Functional Service View. In other words, the Functional Service View does not seem consistent with the Business Operation View for some existing PIPs.

B.7 IFV and Agent/Service References

RNIF 2.0 specifies how the Implementation Framework View of a PIP, with the exception of DTDs and Message Guidelines for business documents, can be derived consistently from the Business Operation View and Functional Service View portions of the PIP specification. Therefore, "boiler-plated" materials related to the Implementation Framework View, including reference to agent/service interactions, should be removed.

APPENDIX C IFV MAPPING FROM BOV AND FSV

This appendix serves to remove "boilerplate" material from the individual PIP specifications and place it in the RNIF. This will facilitate maintenance of this material, as well as remove material from the PIP specifications that is rarely referenced by PIP implementers.

A RosettaNet Partner Interface Process (PIP) specification comprises the following three views of the e-Business PIP model.

1. **Business Operational View (BOV).** Captures the semantics of business data entities and their flow of exchange between roles as they perform business activities. The content of the BOV section is based on the PIP Blueprint document created for RosettaNet's business community.

2. **Functional Service View (FSV).** Specifies the network component services and agents and the interactions necessary to execute PIPs. The FSV includes all of the transaction dialogs in a PIP Protocol. The purpose of the FSV is to specify a PIP Protocol that is systematically derived from the BOV. The two major components within the FSV are the network component design and network component interactions.

3. **Implementation Framework View (IFV).** The Implementation Framework View specifies the action message formats and communication requirements between network components as supported by the RosettaNet Implementation Framework. The communication requirements include specifications on requirement for secure transport protocols such as SSL and digital signatures. For message formats, RosettaNet distributes XML DTDs and Message Guidelines for the action messages that are exchanged when the PIP is executed.

The RNIF 2.0 PIP specifications include the BOV and FSV specifications and the XML Message Guidelines part of the IFV. However, other aspects of IFV such as the communications requirements between network components are no longer specified as part of the PIP specification, as these aspects can be derived from the BOV and FSV parts of the PIP specification in a well-defined and consistent fashion. This appendix describes how the BOV and FSV sections of a PIP specification can be mapped to such Implementation Framework View (IFV) aspects.

In the following tables, the BOV and FSV columns, their values and the corresponding IFV mapping is listed. Table 8~~Table 8~~~~Table 8~~~~Table 8~~~~Table 8~~~~Table 8~~~~Table 7~~~~Table 6~~ contains mappings that are transport independent and Table 9~~Table 9~~~~Table 9~~~~Table 9~~~~Table 9~~~~Table 9~~~~Table 9~~~~Table 8~~~~Table 7~~ contains mappings that are transport dependent.

Note: Please note that the PIP specification table numbers referenced below are consistent with all the PIP specifications published so far. This numbering scheme is expected to continue. However, if the scheme ever changes, this appendix needs to be updated to be consistent with the PIP specifications.

~~Table 6.~~Table 8. Transport-Independent Mappings

BOV Table 3-3 Business Activity Performance Control		FSV Tables 4.3 … 4.n Message Exchange Controls		IFV Mapping
Column Name	**Value**	**Column Name**	**Value**	**Transport-Independent Mapping**
Acknowledgment of Receipt: Non-Repudiation Required?	Y	Is Non-Repudiation Required?	Y	A signed Receipt-Acknowledgment is required for the received RosettaNet Business Message. The Acknowledgment MUST include MD5 or SHA-1 digest of the received message, in addition to the digital signature. Additionally the partner receiving the acknowledgment MUST store the receipt in original form for a mutually agreed period of time (typically three to seven years). This prevents a responding partner later denying that they received a Business Document. Note: Signals are not acknowledged. Hence this is applicable to Action Messages only.
Acknowledgment of Receipt: Time to Acknowledge	>0	Time To Acknowledge Receipt Signal	>0	A Receipt Acknowledgment for the received RosettaNet Business Message is required and MUST be received by the sender within the time constraint specified. However there is no non-repudiation requirement unless specified with a separate non-repudiation clause as above. Note: Signals are not acknowledged. Hence this is applicable to Action Messages only.
Time to Acknowledge Acceptance	N/A	Time To Acknowledge Acceptance Signal	N/A	The Acceptance Acknowledgment Signal had been eliminated by the RNIF 2.0 specification. Hence these columns are no longer needed and would be eliminated from future versions of the PIP Specifications.

BOV Table 3-3 Business Activity Performance Control		FSV Tables 4.3 … 4.n Message Exchange Controls		IFV Mapping
Column Name	**Value**	**Column Name**	**Value**	**Transport-Independent Mapping**
Time to Perform	Value	Time to Respond to Action	Value	The response Business Action Message to the received Business Action Message MUST be sent within the time constraint specified. Note: Certain Action Messages do not require a response Action Message (PIP specific). For such PIPs this field would have a value of N/A.
Is Authorization Required?	Y	Is Authorization Required?	Y	Sender MUST be Authorized to send this RosettaNet Business Message (or perform this business action). Digital Signature is required on the Message, which would be used by the receiving party to authenticate the sender and verify authorization to send the message.
Non-Repudiation of Origin and Content?	Y	Is Non-Repudiation Required?	Y	The partner receiving the RosettaNet Business Message MUST store the message in original form for a mutually agreed period of time (typically three to seven years). This prevents an initiating partner later denying that they originated contents of a Business Document.
Retry Count*	Value			Specified the retry count for the Action Messages within the PIP.

* Retry Count will be moving from BOV to FSV in PIPs adhering to the PIP metamodel arising out of RNIF 2.0. See also Appendix B.

~~Table 7.~~Table 9. Transport-Dependent Mappings

BOV Figure 3-1 PIP Business Process Flow Diagram	FSV Tables 4.3 … 4.n Message Exchange Controls		IFV Mapping		
	Column	**Value**	**General**	**HTTP Transport**	**SMTP Transport**
Business Activity contains <<SecureFlow>> Stereotype	Is Secure Transport Required?	Y	The Business message MUST be transported from sender to the recipient in a secure way.	SSL is required	Message MUST be encrypted during transport.

	Is Persistent Encryption Required?	Y	The Business Message or Signal MUST be secured from end-to-end (originator to final recipient), not only from point-to-point.	Message MUST be encrypted before being transported.	Message MUST be encrypted before being transported.

APPENDIX D IMPORTANCE OF TRANSFER INDEPENDENCE

It is important to understand the reasons for embracing and requiring transfer-level independence.

Transfer independence allows for rapid integration into existing products and systems by allowing the RosettaNet Business Message to be submitted to these systems for additional packaging and transport. Some transfer mechanisms may not be considered robust enough, secure enough, flexible enough, or easy enough to use for every implementation scenario. The following examples should make this clearer.

A corporation may have an existing infrastructure in place for secure communications and wish to leverage this investment for RosettaNet Partner Interface Processes. This existing infrastructure may be a Virtual Private Network (VPN), a secure tunneling infrastructure, an IP-SEC infrastructure, etc. In cases such as this, transfer-level security alone may be sufficient to protect the RosettaNet Business Message.

Because of the very poor Internet infrastructure that exists in some geographies, a trading partner may choose to compress RosettaNet Business Message(s) together or break up large business messages and transport them using HTTP, secure FTP, secure email solutions such as PGP, etc.

A trading entity may need to trade documents over a medium or networking protocol where TCP/IP does not exist, or where industry-standard data protection mechanisms are not deemed adequate.

Transfer independence allows for all of these scenarios and many yet unforeseen scenarios.

APPENDIX E ANTICIPATED FUTURES

This appendix describes some of the technologies examined during the development of this document that seemed promising for later versions of the framework, but which are not yet at the point of being production-worthy in the RosettaNet environment. Inclusion of a given technology in this appendix does not guarantee that RosettaNet will adopt it, nor does it promise adoption on a pre-stated timeline if the decision is made in the future to make use of it in RosettaNet specifications.

E.1 Use of XML-Schemas

Currently PIP IFV specifications use XML DTD format to define the structure of the Action messages and use associated guideline specifications to define semantic and integrity constraints. RosettaNet is closely following the W3C XML-Schema draft specifications and when the specifications do become a standard and software implementations that support the schema specifications become available, RosettaNet intends to use the W3 XML-Schema format to specify the Action and Signal messages. It should be noted that this would not impact the physical encoding of the Action or Signal messages but, provides more robust specification of the schemas for these specifications that support more automated schema validation to the extent facilitated by the schema standards.

E.2 Use of XML D-Signature

As mentioned earlier, RNIF 2.0 uses S/MIME for digital signatures. RosettaNet intends to evaluate and consider for utilization in a future RNIF release, the XML-Dsig specification by W3C when the specification becomes a standard.

E.3 XML-Based Packaging

RosettaNet business messages comprise multiple XML and non-XML documents (e.g. attachments) and other components like digital signatures. In RNIF 2.0 RosettaNet uses MIME and S/MIME based packaging schemes for building the RosettaNet business message, as MIME and S/MIME are found to provide the best and probably only solution to the packaging needs of the RosettaNet Business Message. However, if a better standards based packaging scheme, such as a pure XML based packaging scheme does become available, RosettaNet intends to consider that for adaptation in a future version of RNIF. We are not aware of any potential solution at this point.

E.4 Other Transport (Transfer) Protocols

RNIF 2.0 specification supports the use of HTTP(S) and SMTP protocols for exchange of RosettaNet Business Messages. However, the RosettaNet Business Message format is really transfer protocol independent and hence RosettaNet intends to support other transfer protocols in the future, as needed by the RosettaNet member

community and based on the usability of the transfer protocol for RosettaNet purposes.

E.5 PIP Message Exchange Models

Current PIP specifications are based on a Peer-to-Peer business message exchange model between the RosettaNet networked applications (and hence the trading partners). This peer-peer mode of message exchange requires prior knowledge of the peer, and it does not support broadcast to several trading partners. RosettaNet is investigating other message exchange models such as Publish and Subscribe, Broadcast and Multicast for potential future incorporation into the PIP specifications.

E.6 Grouping Multiple Action Messages

The RNIF 2.0 team considered grouping and packaging schemes that would permit exchanging two or more Action message in a group between trading partners. However, the robust reliable message delivery mechanism that RosettaNet employs, based on different kinds of acknowledgments being exchanged and the associated timeout and message retry constraints made the grouping scheme too complex to use. Additionally it was felt that the grouping scheme was intended for bulk exchange of messages, where a separate network connection for each transferred message was considered too expensive and predates the current HTTP and such recent transfer technologies. The complexities introduced by such grouping scheme outweighed the benefits offered and hence it was decided not to introduce a grouping scheme in RNIF 2.0.

E.7 Non-Repudiation of Routing for Hub-Routed Messages

A complete specification for non-repudiable routing through hubs is planned for a future release of RNIF. In the meantime, hubs are responsible for solving this in private ways.

E.8 Agent-Service Transmissions

This document focuses specifically on data transmissions between trading partners' RosettaNet-aware network applications, also known as ***service-to-service*** transmissions. In the future, depending upon the complexities of a particular business activity or new requirements from trading partners, RosettaNet may have to specify additional transmission patterns. This section discusses some possibilities.

Instead of the typical service-to-service transmissions, business activities or trading partners may require the introduction of one or more intermediaries or ***agents*** between services. An agent may be a human at a browser or perhaps a software application simply acting as a proxy that prevents direct communication between trading partners' services.

In any case, the important thing to note is that an agent simply passes information to another agent or service but is, itself, incapable of actually conversing using the RosettaNet protocol. Only *services* can provide direct bi-directional support for the RosettaNet protocol.

The following are examples of possible transmission patterns that RosettaNet may specify in the future.

The vertical bar indicates the separation between trading partners' RosettaNet-aware network applications. Trading Partner A is to the left of the bar, Trading Partner B is to the right.

- Service-Agent-|-Service or Service-|-Agent-Service

  Trading Partner A's service converses with Trading Partner B's service but passes the RosettaNet business message first to a proxy agent which forwards it to Trading Partner B's service. The agent could, of course, be a human at a browser or a software application that adds, subtracts, or normalizes information in the message. From that point, the trading partner services might converse directly with each other or continue to communicate through the agent.

- Service-Agent-|-Agent-Service

  A variation of the transmission pattern above.

- Service-|-Service-Agent

  Trading Partner A's service communicates directly with Trading Partner B's service using the RosettaNet protocol, however Trading Partner B's service digests and passes information from the RosettaNet business message to a backend agent for processing. The agent might interact with a backend system and return its results to Trading Partner B's service to communicate back to Trading Partner A's service.

- Agent-Service-|-Service

  An agent on Trading Partner A's side (may be a human at a browser) sends information to Trading Partner A's service which then communicates with Trading Partner B's service using the RosettaNet protocol.

E.9 Dynamic Negotiation of Quality of Service Parameters

RNIF 2.0 added a Quality of Service element to the Service Header as a placeholder hook for specifying dynamically negotiable Quality of Service parameters for the message exchange between trading partners. This is a placeholder at this point (for future backward compatibility), to be specified fully in a future version of the RNIF specification.

APPENDIX F ADDITIONAL EXAMPLES

F.1 Complete Unsigned Message-Packaging Example

```
MIME-version: 1.0
Content-Type: multipart/related; boundary="example-boundary";
              type="application/xml"
Content-Description: This is the RosettaNet Business Message

--example-boundary
Content-Type: Application/XML
Content-Location: "RN-Preamble"
Content-ID: <PreambleHdrExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Preamble SYSTEM "Preamble_MS_BV02_00.dtd">
<Preamble>
<standardName>
   <GlobalAdministeringAuthorityCode>RosettaNet</GlobalAdministering
AuthorityCode>
</standardName>
<standardVersion>
   <VersionIdentifier>V02.00</VersionIdentifier>
</standardVersion>
</Preamble>

--example-boundary
Content-Type: Application/XML
Content-Location: "RN-Delivery-Header"
Content-ID: <DeliveryHdrExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DeliveryHeader SYSTEM "DeliveryHeader_MS_BV02_00.dtd">
<DeliveryHeader>
<isSecureTransportRequired>
   <AffirmationIndicator>~~y~~Yes</AffirmationIndicator>
</isSecureTransportRequired>
<messageDateTime>
   <DateTimeStamp>20001121T145200.000Z</DateTimeStamp>
</messageDateTime>
<messageReceiverIdentification>
   <PartnerIdentification>
      <domain>
         <FreeFormText>DUNS</FreeFormText>
      </domain>
      <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      <locationID>
         <~~FreeFormText~~Value>Santa Clara</~~FreeFormText~~Value>
      </locationID>
   </PartnerIdentification>
</messageReceiverIdentification>
<messageSenderIdentification>
   <PartnerIdentification>
      <GlobalBusinessIdentifier>555123456</GlobalBusinessIdentifier>
      <locationID>
         <~~FreeFormText~~Value>Hong Kong</~~FreeFormText~~Value>
      </locationID>
```

```
   </PartnerIdentification>
</messageSenderIdentification>
<messageTrackingID>
   <InstanceIdentifier>543543</InstanceIdentifier>
</messageTrackingID>
</DeliveryHeader>

--example-boundary
Content-Type: Application/XML
Content-Location: "RN-Service-Header"
Content-Description: RosettaNet-Service-Header
Content-ID: <ServiceHdrExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceHeader SYSTEM "ServiceHeader_MS_BV02_00.dtd">
<ServiceHeader>
<ProcessControl>
   <ActivityControl>
      <BusinessActivityIdentifier>Create Purchase
Order</BusinessActivityIdentifier>
      <MessageControl>
         <fromRole>
            <GlobalPartnerRoleClassificationCode>Buyer</GlobalPartner
RoleClassificationCode>
         </fromRole>
         <fromService>
            <GlobalBusinessServiceCode>Buyer
Service</GlobalBusinessServiceCode>
         </fromService>
         <Manifest>
            <Attachment>
               <description>
                  <FreeFormText>PDF version of PO</FreeFormText>
               </description>

<GlobalMimeTypeQualifierCode>PDFapplication/pdf</GlobalMimeType
QualifierCode>
               <UniversalResourceIdentifier>"cid:Attachment.
20001121T123000.000Z@this.example.com"</UniversalResourceIdentifier>
            </Attachment>
            <numberOfAttachments>
               <CountableAmount>1</CountableAmount>
            </numberOfAttachments>
            <ServiceContentControl>
               <ActionIdentity>
                  <GlobalBusinessActionCode>Purchase Order Request
Action</GlobalBusinessActionCode>
                  <VersionIdentifier>01.02</VersionIdentifier>
               </ActionIdentity>
            </ServiceContentControl>
         </Manifest>
         <toRole>
            <GlobalPartnerRoleClassificationCode>Seller</GlobalPartner
RoleClassificationCode>
         </toRole>
         <toService>
            <GlobalBusinessServiceCode>Seller
Service</GlobalBusinessServiceCode>
         </toService>
      </MessageControl>
   </ActivityControl>
   <GlobalUsageCode>Production</GlobalUsageCode>
   <KnownInitiatingPartner>
```

```
      <PartnerIdentification>
         <domain>
            <FreeFormText>DUNS</FreeFormText>
         </domain>
         <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      </PartnerIdentification>
   </KnownInitiatingPartner>
   <pipCode>
      <GlobalProcessIndicatorCode>3A4</GlobalProcessIndicatorCode>
   </pipCode>
   <pipInstanceId>
      <InstanceIdentifier>121212</InstanceIdentifier>
   </pipInstanceId>
   <pipVersion>
      <VersionIdentifier>01.02</VersionIdentifier>
   </pipVersion>
   <KnownInitiatingPartner>
      <PartnerIdentification>
         <domain>
            <FreeFormText>DUNS</FreeFormText>
         </domain>
         <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      </PartnerIdentification>
   </KnownInitiatingPartner>
</ProcessControl>
</ServiceHeader>

--example-boundary
Content-Type: Application/XML
Content-Description: RosettaNet-Service-Content
Content-Location: "RN-Service-Content"
Content-ID:
<ServiceContentExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Pip3A4PurchaseOrderRequest SYSTEM
"3A4PurchaseOrderRequestMessageGuideline_v1_2.dtd">
<Pip3A4PurchaseOrderRequest>
<PurchaseOrder>
   <deliverTo>
      <PhysicalAddress>
         <cityName>
            <FreeFormText xml:lang="EN">CityName</FreeFormText>
         </cityName>
         <addressLine1>
            <FreeFormText xml:lang="EN">1234 Address
Drive</FreeFormText>
         </addressLine1>
         <regionName>
            <FreeFormText xml:lang="EN">Eastern US</FreeFormText>
         </regionName>
         <postOfficeBoxIdentifier>
            <FreeFormText xml:lang="EN">20202</FreeFormText>
         </postOfficeBoxIdentifier>

<GlobalLocationIdentifier>1234567890000</GlobalLocationIdentifier>
         <GlobalCountryCode>US</GlobalCountryCode>
      </PhysicalAddress>
   </deliverTo>
   <ProductLineItem>
      <shipFrom>

<GlobalLocationIdentifier>9876543210000</GlobalLocationIdentifier>
```



```
      </shipFrom>
      <ProductQuantity>1</ProductQuantity>
      <LineNumber>1</LineNumber>
      <productUnit>
         <ProductPackageDescription>
            <ProductIdentification>

<GlobalProductIdentifier>12345678901234</GlobalProductIdentifier>
            </ProductIdentification>
         </ProductPackageDescription>
      </productUnit>
      <countryOfOrigin>
         <GlobalCountryCode>US</GlobalCountryCode>
      </countryOfOrigin>
      <requestedShipDate>
         <DateStamp>20001121</DateStamp>
      </requestedShipDate>
      <contractIdentifier>

<ProprietaryDocumentIdentifier>1021358129419</ProprietaryDocument
Identifier>
      </contractIdentifier>
      <GlobalProductUnitOfMeasureCode>Each</GlobalProductUnitOfMeasure
Code>
      <SpecialHandlingInstruction>
         <specialHandlingText>
            <FreeFormText xml:lang="EN">Hand deliver</FreeFormText>
         </specialHandlingText>
      </SpecialHandlingInstruction>
      <requestedPrice>
         <FinancialAmount>
            <GlobalCurrencyCode>USD</GlobalCurrencyCode>
            <MonetaryAmount>25</MonetaryAmount>
         </FinancialAmount>
      </requestedPrice>
   </ProductLineItem>
   <GlobalShipmentTermsCode>Third party pay</GlobalShipmentTermsCode>
   <RevisionNumber>11</RevisionNumber>
   <prePaymentCheckNumber>
      <CheckNumber>10101</CheckNumber>
   </prePaymentCheckNumber>
   <QuoteIdentifier>

<ProprietaryDocumentIdentifier>12345</ProprietaryDocumentIdentifier>
   </QuoteIdentifier>
   <WireTransferIdentifier>88888</WireTransferIdentifier>
   <AccountDescription>
      <GlobalAccountClassificationCode>Procurement</GlobalAccount
ClassificationCode>
      <billTo>
         <PartnerRoleDescription>

<GlobalPartnerRoleClassificationCode>Buyer</GlobalPartnerRole
ClassificationCode>
            <ContactInformation>
               <PhysicalAddress>
                  <cityName>
                     <FreeFormText xml:lang="EN">City Name</FreeFormText>
                  </cityName>
                  <addressLine1>
                     <FreeFormText xml:lang="EN">3877 Fairfax Ridge Rd,
4th
Floor</FreeFormText>
```

```
                  </addressLine1>
                  <addressLine2>
                     <FreeFormText xml:lang="EN">Fairfax, VA
22030</FreeFormText>
                  </addressLine2>
                  <regionName>
                     <FreeFormText xml:lang="EN">Eastern
US</FreeFormText>
                  </regionName>
                  <postOfficeBoxIdentifier>
                     <FreeFormText xml:lang="EN">20202</FreeFormText>
                  </postOfficeBoxIdentifier>

<GlobalLocationIdentifier>9876543210000</GlobalLocation
Identifier>
                  <GlobalCountryCode>US</GlobalCountryCode>
               </PhysicalAddress>
               <EmailAddress>contact@rnifexample.com</EmailAddress>
               <contactName>
                  <FreeFormText xml:lang="EN">Mr. Contact
Smith</FreeFormText>
               </contactName>
               <telephoneNumber>
                  <CommunicationsNumber>555-555-
5555</CommunicationsNumber>
               </telephoneNumber>
            </ContactInformation>
         </PartnerRoleDescription>
      </billTo>
      <accountName>
         <FreeFormText xml:lang="EN">Cash Account</FreeFormText>
      </accountName>
      <AccountNumber>12341234</AccountNumber>
   </AccountDescription>
   <generalServicesAdministrationNumber>
      <ProprietaryDocumentIdentifier>11111111</ProprietaryDocument
Identifier>
   </generalServicesAdministrationNumber>
   <GlobalFinanceTermsCode>Net 30</GlobalFinanceTermsCode>
   <PartnerDescription>
      <PhysicalAddress>
         <cityName>
            <FreeFormText xml:lang="EN"/>
         </cityName>
         <addressLine1>
            <FreeFormText xml:lang="EN">1234 Address
Drive</FreeFormText>
         </addressLine1>
         <regionName>
            <FreeFormText xml:lang="EN">Eastern US</FreeFormText>
         </regionName>
         <postOfficeBoxIdentifier>
            <FreeFormText xml:lang="EN">20202</FreeFormText>
         </postOfficeBoxIdentifier>
         <GlobalCountryCode>US</GlobalCountryCode>
      </PhysicalAddress>
      <GlobalPartnerClassificationCode>End
User</GlobalPartnerClassificationCode>
   </PartnerDescription>
   <GlobalPurchaseOrderTypeCode>Dropship</GlobalPurchaseOrderTypeCode>
</PurchaseOrder>
<fromRole>
   <PartnerRoleDescription>
```



```
      <GlobalPartnerRoleClassificationCode>Buyer</GlobalPartnerRole
ClassificationCode>
      <ContactInformation>
         <EmailAddress>xyz@abc.com</EmailAddress>
         <contactName>
            <FreeFormText xml:lang="EN">Somebody</FreeFormText>
         </contactName>
         <telephoneNumber>
            <CommunicationsNumber>888-888-8888</CommunicationsNumber>
         </telephoneNumber>
      </ContactInformation>
      <PartnerDescription>
         <GlobalPartnerClassificationCode>End
User</GlobalPartnerClassificationCode>
         <BusinessDescription>

<GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
            <GlobalSupplyChainCode>Information
Technology</GlobalSupplyChainCode>
         </BusinessDescription>
      </PartnerDescription>
   </PartnerRoleDescription>
</fromRole>
<toRole>
   <PartnerRoleDescription>
      <GlobalPartnerRoleClassificationCode>Seller</GlobalPartnerRole
ClassificationCode>
      <PartnerDescription>
         <GlobalPartnerClassificationCode>End
User</GlobalPartnerClassificationCode>
         <BusinessDescription>

<GlobalBusinessIdentifier>987654321</GlobalBusinessIdentifier>
            <GlobalSupplyChainCode>Information
Technology</GlobalSupplyChainCode>
         </BusinessDescription>
      </PartnerDescription>
   </PartnerRoleDescription>
</toRole>
<thisDocumentGenerationDateTime>
   <DateTimeStamp>20001121T080010.005Z</DateTimeStamp>
</thisDocumentGenerationDateTime>
<thisDocumentIdentifier>
   <ProprietaryDocumentIdentifier>1021358129419</ProprietaryDocument
Identifier>
</thisDocumentIdentifier>
<GlobalDocumentFunctionCode>Request</GlobalDocumentFunctionCode>
</Pip3A4PurchaseOrderRequest>

--example-boundary
Content-Type: application/pdf; name="PO.pdf"
Content-Description: PDF version of PO
Content-ID: <Attachment.20001121T123000.000Z@this.example.com>

[PO.pdf attachment goes here]

--example-boundary--
```

F.2 Complete Signed Message-Packaging Example

```
MIME-version: 1.0
```

```
Content-Type: multipart/signed;
              boundary="RN-Signature-Boundary";
              protocol="application/pkcs7-signature";
              micalg=sha1
Content-Description: This is a Signed RosettaNet Business Message

--RN-Signature-Boundary
Content-Type: multipart/related; boundary="example-boundary";
              type="application/xml"
Content-Description: This is the RosettaNet Business Message

--example-boundary
Content-Type: Application/XML
Content-Location: "RN-Preamble"
Content-ID: <PreambleHdrExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Preamble SYSTEM "Preamble_MS_BV02_00.dtd">
<Preamble>
<standardName>
   <GlobalAdministeringAuthorityCode>RosettaNet</GlobalAdministering
AuthorityCode>
</standardName>
<standardVersion>
   <VersionIdentifier>V02.00</VersionIdentifier>
</standardVersion>
</Preamble>

--example-boundary
Content-Type: Application/XML
Content-Location: "RN-Delivery-Header"
Content-ID: <DeliveryHdrExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DeliveryHeader SYSTEM "DeliveryHeader_MS_BV02_00.dtd">
<DeliveryHeader>
<isSecureTransportRequired>
   <AffirmationIndicator>yYes</AffirmationIndicator>
</isSecureTransportRequired>
<messageDateTime>
   <DateTimeStamp>20001121T145200.000Z</DateTimeStamp>
</messageDateTime>
<messageReceiverIdentification>
   <PartnerIdentification>
      <domain>
         <FreeFormText>DUNS</FreeFormText>
      </domain>
      <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      <locationID>
         <FreeFormTextValue>Santa Clara</FreeFormTextValue>
      </locationID>
   </PartnerIdentification>
</messageReceiverIdentification>
<messageSenderIdentification>
   <PartnerIdentification>
      <GlobalBusinessIdentifier>555123456</GlobalBusinessIdentifier>
      <locationID>
         <FreeFormTextValue>Hong Kong</FreeFormTextValue>
      </locationID>
   </PartnerIdentification>
</messageSenderIdentification>
<messageTrackingID>
   <InstanceIdentifier>543543</InstanceIdentifier>
```



```
</messageTrackingID>
</DeliveryHeader>

--example-boundary
Content-Type: Application/XML
Content-Location: "RN-Service-Header"
Content-Description: RosettaNet-Service-Header
Content-ID: <ServiceHdrExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceHeader SYSTEM "ServiceHeader_MS_BV02_00.dtd">
<ServiceHeader>
<ProcessControl>
   <ActivityControl>
      <BusinessActivityIdentifier>Create Purchase
Order</BusinessActivityIdentifier>
      <MessageControl>
         <fromRole>
            <GlobalPartnerRoleClassificationCode>Buyer</GlobalPartner
RoleClassificationCode>
         </fromRole>
         <fromService>
            <GlobalBusinessServiceCode>Buyer
Service</GlobalBusinessServiceCode>
         </fromService>
         <Manifest>
            <Attachment>
               <description>
                  <FreeFormText>PDF version of PO</FreeFormText>
               </description>

<GlobalMimeTypeQualifierCode>PDFapplication/pdf</GlobalMimeType
QualifierCode>
               <UniversalResourceIdentifier>"cid:Attachment.
20001121T123000.000Z@this.example.com"</UniversalResourceIdentifier>
            </Attachment>
            <numberOfAttachments>
               <CountableAmount>1</CountableAmount>
            </numberOfAttachments>
            <ServiceContentControl>
               <ActionIdentity>
                  <GlobalBusinessActionCode>Purchase Order Request
Action</GlobalBusinessActionCode>
                  <VersionIdentifier>01.02</VersionIdentifier>
               </ActionIdentity>
            </ServiceContentControl>
         </Manifest>
         <toRole>
            <GlobalPartnerRoleClassificationCode>Seller</GlobalPartner
RoleClassificationCode>
         </toRole>
         <toService>
            <GlobalBusinessServiceCode>Seller
Service</GlobalBusinessServiceCode>
         </toService>
      </MessageControl>
   </ActivityControl>
   <GlobalUsageCode>Production</GlobalUsageCode>
   <KnownInitiatingPartner>
      <PartnerIdentification>
         <domain>
            <FreeFormText>DUNS</FreeFormText>
         </domain>
```

```
~~      <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>~~
~~    </PartnerIdentification>~~
~~  </KnownInitiatingPartner>~~
   <pipCode>
      <GlobalProcessIndicatorCode>3A4</GlobalProcessIndicatorCode>
   </pipCode>
   <pipInstanceId>
      <InstanceIdentifier>121212</InstanceIdentifier>
   </pipInstanceId>
   <pipVersion>
      <VersionIdentifier>~~0~~1.~~0~~2</VersionIdentifier>
   </pipVersion>
   <KnownInitiatingPartner>
      <PartnerIdentification>
         <domain>
            <FreeFormText>DUNS</FreeFormText>
         </domain>
         <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
      </PartnerIdentification>
   </KnownInitiatingPartner>
</ProcessControl>
</ServiceHeader>

--example-boundary
Content-Type: Application/XML
Content-Description: RosettaNet-Service-Content
Content-Location: "RN-Service-Content"
Content-ID:
<ServiceContentExample.20001121T123100.000Z@this.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Pip3A4PurchaseOrderRequest SYSTEM
"3A4PurchaseOrderRequestMessageGuideline_v1_2.dtd">
<Pip3A4PurchaseOrderRequest>
<PurchaseOrder>
   <deliverTo>
      <PhysicalAddress>
         <cityName>
            <FreeFormText xml:lang="EN">CityName</FreeFormText>
         </cityName>
         <addressLine1>
            <FreeFormText xml:lang="EN">1234 Address
Drive</FreeFormText>
         </addressLine1>
         <regionName>
            <FreeFormText xml:lang="EN">Eastern US</FreeFormText>
         </regionName>
         <postOfficeBoxIdentifier>
            <FreeFormText xml:lang="EN">20202</FreeFormText>
         </postOfficeBoxIdentifier>

<GlobalLocationIdentifier>1234567890000</GlobalLocationIdentifier>
         <GlobalCountryCode>US</GlobalCountryCode>
      </PhysicalAddress>
   </deliverTo>
   <ProductLineItem>
      <shipFrom>

<GlobalLocationIdentifier>9876543210000</GlobalLocationIdentifier>
      </shipFrom>
      <ProductQuantity>1</ProductQuantity>
      <LineNumber>1</LineNumber>
      <productUnit>
```



```
            <ProductPackageDescription>
               <ProductIdentification>

<GlobalProductIdentifier>12345678901234</GlobalProductIdentifier>
               </ProductIdentification>
            </ProductPackageDescription>
         </productUnit>
         <countryOfOrigin>
            <GlobalCountryCode>US</GlobalCountryCode>
         </countryOfOrigin>
         <requestedShipDate>
            <DateStamp>20001121</DateStamp>
         </requestedShipDate>
         <contractIdentifier>

<ProprietaryDocumentIdentifier>1021358129419</ProprietaryDocument
Identifier>
         </contractIdentifier>

<GlobalProductUnitOfMeasureCode>Each</GlobalProductUnitOfMeasureCode>
         <SpecialHandlingInstruction>
            <specialHandlingText>
               <FreeFormText xml:lang="EN">Hand deliver</FreeFormText>
            </specialHandlingText>
         </SpecialHandlingInstruction>
         <requestedPrice>
            <FinancialAmount>
               <GlobalCurrencyCode>USD</GlobalCurrencyCode>
               <MonetaryAmount>25</MonetaryAmount>
            </FinancialAmount>
         </requestedPrice>
      </ProductLineItem>
      <GlobalShipmentTermsCode>Third party pay</GlobalShipmentTermsCode>
      <RevisionNumber>11</RevisionNumber>
      <prePaymentCheckNumber>
         <CheckNumber>10101</CheckNumber>
      </prePaymentCheckNumber>
      <QuoteIdentifier>

<ProprietaryDocumentIdentifier>12345</ProprietaryDocumentIdentifier>
      </QuoteIdentifier>
      <WireTransferIdentifier>88888</WireTransferIdentifier>
      <AccountDescription>
         <GlobalAccountClassificationCode>Procurement</GlobalAccount
ClassificationCode>
         <billTo>
            <PartnerRoleDescription>

<GlobalPartnerRoleClassificationCode>Buyer</GlobalPartnerRole
ClassificationCode>
               <ContactInformation>
                  <PhysicalAddress>
                     <cityName>
                        <FreeFormText xml:lang="EN">City Name</FreeFormText>
                     </cityName>
                     <addressLine1>
                        <FreeFormText xml:lang="EN">3877 Fairfax Ridge Rd,
4th Floor</FreeFormText>
                     </addressLine1>
                     <addressLine2>
                        <FreeFormText xml:lang="EN">Fairfax, VA
22030</FreeFormText>
                     </addressLine2>
```

```
                  <regionName>
                     <FreeFormText xml:lang="EN">Eastern
US</FreeFormText>
                  </regionName>
                  <postOfficeBoxIdentifier>
                     <FreeFormText xml:lang="EN">20202</FreeFormText>
                  </postOfficeBoxIdentifier>

<GlobalLocationIdentifier>9876543210000</GlobalLocation
Identifier>
                  <GlobalCountryCode>US</GlobalCountryCode>
               </PhysicalAddress>
               <EmailAddress>contact@rnifexample.com</EmailAddress>
               <contactName>
                  <FreeFormText xml:lang="EN">Mr. Contact
Smith</FreeFormText>
               </contactName>
               <telephoneNumber>
                  <CommunicationsNumber>555-555-
5555</CommunicationsNumber>
               </telephoneNumber>
            </ContactInformation>
         </PartnerRoleDescription>
      </billTo>
      <accountName>
         <FreeFormText xml:lang="EN">Cash Account</FreeFormText>
      </accountName>
      <AccountNumber>12341234</AccountNumber>
   </AccountDescription>
   <generalServicesAdministrationNumber>
      <ProprietaryDocumentIdentifier>11111111</ProprietaryDocument
Identifier>
   </generalServicesAdministrationNumber>
   <GlobalFinanceTermsCode>Net 30</GlobalFinanceTermsCode>
   <PartnerDescription>
      <PhysicalAddress>
         <cityName>
            <FreeFormText xml:lang="EN"/>
         </cityName>
         <addressLine1>
            <FreeFormText xml:lang="EN">1234 Address
Drive</FreeFormText>
         </addressLine1>
         <regionName>
            <FreeFormText xml:lang="EN">Eastern US</FreeFormText>
         </regionName>
         <postOfficeBoxIdentifier>
            <FreeFormText xml:lang="EN">20202</FreeFormText>
         </postOfficeBoxIdentifier>
         <GlobalCountryCode>US</GlobalCountryCode>
      </PhysicalAddress>
      <GlobalPartnerClassificationCode>End
User</GlobalPartnerClassificationCode>
   </PartnerDescription>
   <GlobalPurchaseOrderTypeCode>Dropship</GlobalPurchaseOrderTypeCode>
</PurchaseOrder>
<fromRole>
   <PartnerRoleDescription>
      <GlobalPartnerRoleClassificationCode>Buyer</GlobalPartnerRole
ClassificationCode>
      <ContactInformation>
         <EmailAddress>xyz@abc.com</EmailAddress>
         <contactName>
```



```
            <FreeFormText xml:lang="EN">Somebody</FreeFormText>
         </contactName>
         <telephoneNumber>
            <CommunicationsNumber>888-888-8888</CommunicationsNumber>
         </telephoneNumber>
      </ContactInformation>
      <PartnerDescription>
         <GlobalPartnerClassificationCode>End
User</GlobalPartnerClassificationCode>
         <BusinessDescription>

<GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
            <GlobalSupplyChainCode>Information
Technology</GlobalSupplyChainCode>
         </BusinessDescription>
      </PartnerDescription>
   </PartnerRoleDescription>
</fromRole>
<toRole>
   <PartnerRoleDescription>
      <GlobalPartnerRoleClassificationCode>Seller</GlobalPartnerRole
ClassificationCode>
      <PartnerDescription>
         <GlobalPartnerClassificationCode>End
User</GlobalPartnerClassificationCode>
         <BusinessDescription>

<GlobalBusinessIdentifier>987654321</GlobalBusinessIdentifier>
            <GlobalSupplyChainCode>Information
Technology</GlobalSupplyChainCode>
         </BusinessDescription>
      </PartnerDescription>
   </PartnerRoleDescription>
</toRole>
<thisDocumentGenerationDateTime>
   <DateTimeStamp>20001121T080010.005Z</DateTimeStamp>
</thisDocumentGenerationDateTime>
<thisDocumentIdentifier>
   <ProprietaryDocumentIdentifier>1021358129419</ProprietaryDocument
Identifier>
</thisDocumentIdentifier>
<GlobalDocumentFunctionCode>Request</GlobalDocumentFunctionCode>
</Pip3A4PurchaseOrderRequest>

--example-boundary
Content-Type: application/pdf; name="PO.pdf"
Content-Description: PDF version of PO
Content-ID: <Attachment.20001121T123000.000Z@this.example.com>

[PO.pdf attachment goes here]

--example-boundary--

--RN-Signature-Boundary
Content-Type: Application/pkcs7-signature; name="detached.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
Content-Description: This is the signature for the Business Message

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
```

```
--RN-Signature-Boundary--
```


APPENDIX G REFERENCES

- *Understanding a PIP Blueprint*. RosettaNet, 1999, 2000. (Source: http://www.rosettanet.org)
- *RosettaNet Technical Conventions and Style Guide.* RosettaNet, 2000. (Source: http://www.rosettanet.org)
- RosettaNet PIP Specifications (complete collection). RosettaNet, 1998-ongoing. (Source: http://www.rosettanet.org)
- RosettaNet Dictionaries (business, technical). RosettaNet, 1998-ongoing. (Source: http://www.rosettanet.org)
- RFC 822: "Standard for the format of ARPA Internet text messages." David H. Crocker. 1982. (Source: ftp://ftp.isi.edu/in-notes/rfc822.txt)
- RFC 1891: "1996 SMTP Service Extension for Delivery Status Notifications." K. Moore. IETF, Network Working Group, 1996. (Source: http://www.ietf.org/rfc/rfc1896.txt)
- RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies." N. Freed, et al. IETF, Network Working Group, 1996. (Source: http://www.ietf.org/rfc/rfc2045.txt)
- RFC 2046: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types." N. Freed, et al. IETF, Network Working Group, 1996. (Source: http://www.ietf.org/rfc/rfc2046.txt)
- RFC 2047: "Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text." K. Moore. IETF, Network Working Group, 1996. (Source: http://www.ietf.org/rfc/rfc2047.txt)
- RFC 2048: "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures." N. Freed, et al. IETF, Network Working Group, 1996. (Source: http://www.ietf.org/rfc/rfc2048.txt)
- RFC 2049: "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples." N. Freed, et al. IETF, Network Working Group, 1996. (Source: http://www.ietf.org/rfc/rfc2049.txt)
- RFC 2111: "Content-ID and Message-ID Uniform Resource Locators." E. Levinson. IETF, Network Working Group, 1997 (Source: http://www.ietf.org/rfc/rfc2111.txt)
- RFC 2119: "Key Words for Use in RFCs to Indicate Requirement Levels." S. Bradner. IETF, Network Working Group, 1997. (Source: http://www.ietf.org/rfc/rfc2119.txt)

- RFC 2311: "S/MIME Version 2 Message Specification." S. Dusse, et al. IETF, Network Working Group, 1998. (Source: http://www.ietf.org/rfc/rfc2311.txt)
- RFC 2312: "S/MIME Version 2 Certificate Handling" S. Dusse, et al. IETF, Network Working Group, 1998. (Source: http://www.ietf.org/rfc/rfc2312.txt)
- RFC 2376: "XML Media Types" E. Whitehead, et al. IETF, Network Working Group, 1998. (Source: http://www.ietf.org/rfc/rfc2376.txt)
- RFC 2387: "The MIME Multipart/Related Content-type." E. Levinson. IETF, Network Working Group, 1998. (Source: http://www.ietf.org/rfc/rfc2387.txt)
- RFC 2557: "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)." J. Palme, et al. IETF, Network Working Group, 1999. (Source: http://www.ietf.org/rfc/rfc2557.txt)
- RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1." R. Fielding, et al. IETF, Network Working Group, 1999. (Source: http://www.ietf.org/rfc/rfc2616.txt)
- ISO 8601:1988. Data elements and interchange formats -- Information interchange -- Representation of dates and times. (Available from: http://www.iso.ch/cate/cat.html)
- *PKCS #7: Cryptographic Message Syntax Standard*. An RSA Laboratories Technical Note. Version 1.5. Revised November 1, 1993, and PKCS-7 version 1.6 bulletin: *Extensions and Revisions to PKCS #7* (13 May 1997): Source: http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs-7.html (January 5, 1999)
- *S/MIME Implementation Guide,* Version 2. Steve Dusse, RSA Labs, ©1996. (Source: http://www.rsa.com)
- Recommendation X.208 (11/88) "Specification of Abstract Syntax Notation One (ASN.1)" (Technically aligned with ISO 8824.) ITU-T (formerly CCITT). (Available at: http://www.itu.int)
- Recommendation X.209 (11/88) "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)" ITU-T (formerly CCITT). (Available at: http://www.itu.int)
- Recommendation X.509 (08/97) - Information technology - Open Systems Interconnection - The Directory: Authentication framework. (Available at: http://www.itu.int/itudoc/itu-t/rec/obsolete/x/x500up/x509_97.html)
- *Extensible Markup Language (XML) 1.0*, W3C Recommendation. Tim Bray, Jean Paoli, C.M. Sperberg-McQueen. Worldwide Web Consortium (W3C), February 1998. (Source: www.w3.org/TR/REC-xml)

APPENDIX H GLOSSARY

action message: a properly packaged business action message. *See also* RosettaNet Business Message.

asynchronous: Communication among distributed processes is said to be "asynchronous" when there is no expectation that the reply to a request comes within the time interval in which the communication session of the request is still "live." Compare with "synchronous."

authorization: permission to access a protected resource, a service, or sensitive information. Sometimes confused with *authentication*, which is simply verification that a user is who he claims to be. One can be properly authenticated but not be authorized to access a protected resource, a service, or sensitive information.

BOV: Business Operational View (concept from ISO 14662 Open-EDI Reference Model). The first section of every PIP specification, the BOV describes the *business-related* aspects of the PIP. This is information captured from business analysts during development of the PIP. The BOV is the PIP Blueprint as approved by the RosettaNet members.

business action: a message with content of a business nature such as a Purchase Order Request or a Request For Quote. The exchange of business actions and business signals comprise the message choreography necessary to complete a business activity specified by a given PIP.

business activity: a PIP encapsulates one or more discrete business activities as specified by the business analysts during development of the PIP blueprint. For example, PIP 3A4 (Manage Purchase Order) specifies three (3) separate business activities: Create Purchase Order, Change Purchase Order, and Cancel Purchase Order. The exchange of business actions and business signals comprise the message choreography necessary to complete a business activity specified by a given PIP.

business message: *see* RosettaNet Business Message.

business signal: a message exchanged between two RosettaNet network applications to communicate certain events *within* the execution of a PIP instance. Examples of signals include "receipt and successful validation of a message" (Receipt Acknowledgment) and "receipt of a message out of sequence" (General Exception). A signal is used to communicate an exception condition *within* the normal message choreography of a PIP. *See also* Process Control PIP.

compliance: an implementation is *compliant* if and only if it fully meets each and every requirement of the RNIF specification. In particular, each and every transaction, action, or data element emitted by the implementation must be valid as defined in "Validation" below. Compliance testing is the act of comparing an implementation's operation against the specified requirements to determine compliance or noncompliance.

conformance: the ability to demonstrate in an unambiguous way that a given implementation is correct with respect to the formal model. (from the Foundation for Intelligent Physical Agents, www.fipa.org/spec/fipa97/fipa97.html)

data element: a basic unit of identifiable and definable data (ISO 10324,1997), a basic unit of data for the purpose of recording and interchange (ISO 2146,1988).

DTD: a type of schema used to specify the structure and semantics of an XML document or message.

e-business: an enterprise that conducts many of its business functions through electronic means. The term also refers to businesses that operate on the Internet and offer goods, services, and information for sale via the Web. (from Jonar C. Nader, Prentice Hall's Illustrated Dictionary of Computing, 3rd edition, 1998)

framework: a set of related architectural components.

FSV: Functional Service View (concept from ISO/IEC 14662 Open-EDI Reference Model). The second section of every PIP specification, the FSV describes the PIP exchange protocol sometimes known as the message choreography or dialog between trading partners during the execution of the PIP. The FSV is systematically derived from the BOV.

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

header: Control information prepended to content.

IFV: Implementation Framework View. The IFV provides the transfer protocol specific requirements for any given PIP, based upon the requirements in the BOV and FSV sections of the PIP, as well as the format of the service content. The mapping of the transfer protocol specific requirements is provided in an appendix ~~in~~ of the RNIF: Core Specification ~~02.00~~, while the format of the service content is packaged with the PIP specification.

implementation framework: guidelines for creating instances of related architectural components.

Manifest: a component of the Service Header that provides information (in the form of a structured listing) about the payload. It describes certain characteristics of the Service Content and also lists the number of attachments included in the payload.

message: a properly packaged business action or business signal. *See also* business action, business signal, and RosettaNet Business Message.

message choreography: the exchange of business actions and business signals required to complete a business activity specified by a given PIP.

message guideline: part of a published RosettaNet specification, a message guideline provides information that supports, but cannot be specified in, a particular declarative schema. Both the message guideline and the declarative schema (presently an XML DTD) are used to validate that a particular message or service content is properly formatted and uses expected values.

non-repudiation: the ability of a message transfer system to provide unforgeable evidence that a specific action occurred. Three types of the non-repudiation services are most common: non-repudiation of origin, non-repudiation of submission, and non-repudiation of delivery. Non-repudiation of origin protects against any attempt by a message originator to deny sending a message. Non-repudiation of submission protects against any attempt by a message transfer agent to deny that a message was submitted for delivery. Non-repudiation of delivery protects against any attempt by a message recipient to deny receiving a message.

one-action activity: a business activity comprised of the following message choreography. Partner A sends a business action to Partner B and Partner B sends a Receipt Acknowledgment signal back to Partner A. When these messages have been exchanged successfully between these trading partners, the activity is deemed complete. PIP 2A1 (Distribute New Product Information) is an example of a PIP that specifies one-action activities.

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: the Service Content plus any file attachments comprises the payload component of a RosettaNet Business Message. The payload is packaged together with the headers to form a complete RosettaNet Business Message.

PIP: *See* Partner Interface Process (PIP).

Preamble Header: an XML document that identifies the name and version of the standard with which the business message is compliant. It is packaged together with other headers and the payload to form a complete RosettaNet Business Message.

Process Control PIP: a type of PIP used to communicate process states *outside* the context of the process instance with which it is associated. For example, PIP 0A1 (Notification of Failure or NoF) is a process control PIP that is used to communicate an exception condition that occurs *outside* the normal message choreography of the subject PIP. *See also* business signal.

protocol: a protocol is a formal set of rules and conventions that governs how computers exchange information over a network medium.

Receipt Acknowledgment: a positive business signal that acknowledges receipt of a message. The Receipt Acknowledgment is sent from the receiver of a *valid* business action message back to the sender. Validity of the message is determined by RNIF base-level validation or by additional validation requirements negotiated between trading partners.

RosettaNet Business Message: the logical grouping of the preamble header, delivery header, service header, and payload (in the case of business action messages).

schema: a specification for the structure and semantics of some related data. One uses the schema to validate or otherwise understand a group of data. One type of schema is the XML-DTD.

service: a networked application that is capable of participating in a RosettaNet conversation.

service message: messages exchanged between services.

Service Content: the primary component of the payload of a RosettaNet Business Message. It is an XML document that represents the business content specified by a particular PIP. The Service Content plus any file attachments comprises the payload component of the RosettaNet Business Message.

Service Header: an XML document that identifies the PIP, the business activity and action with which the business message is associated, the sending and receiving services, partners, roles, etc. It is packaged together with other headers and the payload to form a complete RosettaNet Business Message.

single action activity: *see* one-action activity.

solution partner: An organization or company that produces an RNIF 2.0-compliant product(s).

specification: a detailed formulation, in document form, which provides a definitive description of a system for the purpose of developing or validating the system. [ISO/IEC 2382, Information technology – Vocabulary, 1997]

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.

structure: something composed of organized or interrelated elements; the manner in which the elements of something are organized or interrelated

synchronous: a mode of coordination of communication among distributed processes that requires request-reply pairs to occur within the bounds of some time interval in which the communication session is said to be "live." No implication is made about whether the processes or threads "block" while waiting for a response, though it is assumed that some mechanism of expecting the response within the time interval exists. In practice for internet communication protocols, synchronous communication exists when the reply to a request is conveyed over the same "connection," which for TCP based communication, means that the bounding time interval is that of the TCP connection. Though there are RosettaNet timeouts for replies, these intervals do not involve maintaining a communicative connection throughout and so are not thought of as synchronous with respect to communication primitives.

syntax: the patterns of formation of sentences and phrases from words and the rules for the formation of grammatical sentences in a language.

TPA: *see* Trading Partner Agreement.

trading partner: An organization or company that transacts business using RosettaNet specifications.

Trading Partner Agreement (TPA): information exchanged between trading partners that describes certain mutually agreed upon execution parameters and service level expectations that will be used when conducting business between them.

two-action activity: : a business activity comprised of the following message choreography. Partner A sends a business action to Partner B, Partner B sends a Receipt Acknowledgment signal back to Partner A, some time later Partner B sends a response business action to Partner A, and Partner A sends a Receipt Acknowledgment back to Partner B. When these messages have been exchanged successfully between these trading partners, the activity is deemed complete. PIP 3A4 (Manage Purchase Order) is an example of a PIP that specifies a two-action activity.

valid XML document: An XML document is **valid** if it has an associated document type declaration and if the document complies with the constraints expressed in it. (From World Wide Web Consortium, *Extensible Markup Language (XML) 1.0: W3C Recommendation* 10-February-1998.)

validation: A data element, action, transaction, or process is *valid* if and only if it meets each and every requirement of the RNIF specification, as well as the each and every requirement of the relevant PIP specification. Validation is the act of comparing such an entity against the specified requirements to determine validity or invalidity. Note that each action within a transaction must meet the content and sequence requirements for that transaction. Similarly, each transaction within a process must meet the content and sequence requirements of that process. Such validation is an essential part of testing an implementation. It is also anticipated that the validation team will develop specific requirements for such validation during production use of an implementation.

vocabulary: the collection of words known to a particular person or group and used for a particular purpose.

well-formed XML document: An XML document that, taken as a whole, matches the XML production labeled "document**,"** meets all the well-formedness constraints given in the XML specification, and each of the parsed entities which is referenced directly or indirectly within the document is well-formed. A well-formed document may also be "valid" if it meets additional criteria. (Adapted from World Wide Web Consortium, *Extensible Markup Language (XML) 1.0: W3C Recommendation* 10-February-1998.) (*See also* valid XML document.)

XML document: a data object made up of virtual storage units called entities, which contain 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. (From www.w3.org/TR/PR-xml-971208) *See also* well-formed XML document; valid XML document.