RosettaNet Implementation Framework Specification

Status: Release

Version: 1.1

Date: 8 November 1999

Legal Disclaimer

THE DRAFT SPECIFICATIONS SET FORTH HEREIN ARE FOR DISCUSSION PURPOSES ONLY. THIS IS A WORKING DOCUMENT AND IS NOT INTENDED FOR COMMERCIAL USE OR PUBLIC DISSEMINATION. NEITHER ROSETTANET NOR ITS MEMBERS SHALL BE RESPONSIBLE FOR ANY LOSS RESULTING FROM ANY USE OF THIS DOCUMENT OR THE SPECIFICATIONS HEREIN.

Other Acknowledgements

CONTENTS

Figures

Tables

Examples

VERSION HISTORY

Version 0.1	14 Dec 1998	Initial draft for comment by Project Architects
Version 0.2	18 Dec 1998	Draft revised by Project Architects
Version 0.3	21 Dec 1998	Successive revision
Version 0.4	22 Dec 1998	Draft upgraded with new implementation content.
Version 0.5	23 Dec 1998	Successive revision
Version 0.6	5 Jan 1999	Significant revision
Version 0.7	19 Jan 1999	Final revision by Project Architects
Version 0.8	26 Jan 1999	Incorporating Project Architects final comments
Version 1.0	16 Mar 1999	Incorporating industry feedback
Version 1.01	08 July 1999	Specification Document Change
Version 1.1	08 Nov 1999	Specification Revisions: • Incorporate Section 10 of v1.01 into body of document as appropriate; eliminate section 10 • Complete rewrite of Section 2 and much of Section 3 (eliminate EDI vocabulary & message structures; add XML message header and content structures, including Sequence Validation; add message format; change requirement for base64 encoding) • Alterations to digital signatures • All PIP and protocol examples replaced/updated using PIP 2A8.

Executive Summary

RosettaNet's mission is to harness the global and pervasive reach of the Internet by defining and leading the implementation of open and common electronic business processes between partners in the Information Technology Supply Chain. These processes are designed to align the electronic business interfaces between partners, ultimately resulting in measurable benefits for IT buyers and supply chain partners. RosettaNet specifies these open and common e-business processes as Partner Interface Processes (PIPs) and their implementation guidelines. RosettaNet distributes these guidelines to partners in the supply chain so that they can configure their specific e-business processes to inter-operate with those of their partners.

The RosettaNet Implementation Framework project was chartered to specify an open and common RosettaNet networked-application framework (often abbreviated as "RNIF") so that partners and RosettaNet solution providers can implement computer solutions that can collaboratively execute RosettaNet-compliant PIPs. The following RosettaNet objectives drove the creation of this specification:

- *Streamline Execution*: RosettaNet needs to facilitate the rapid development of PIPs.
- *Accelerate Adoption*: RosettaNet needs to facilitate the rapid development of e-business applications that execute RosettaNet-compliant PIPs.

This document presents a PIP specification model and a networked application framework that allows RosettaNet to meet these objectives. The PIP specification model enables RosettaNet to specify partner-to-partner electronic business processes in terms of "actions," "transactions" and "execution processes." The implementation framework specification enables RosettaNet partners and solution providers to create networked applications that can execute these electronic business processes by communicating according to strictly defined protocols. These protocols specify application message formats and message exchange sequences. Also, this specification includes authentication, authorization, encryption and non-repudiation implementation aspects that are necessary for conducting secure electronic business over the Internet.

As a result of the EConcert pilot programs and RosettaNet research efforts, there have been a number of findings on the current state of the RosettaNet Implementation Framework (RNIF) specifications. These findings ranged from simple clarifications that needed to be included in the current documentation, to erroneous information that was in the documentation, to changes in direction (such as moving from EDI vocabulary and structure and RosettaNet vocabulary and XML-encoding of RosettaNet structures). This version of the document communicates changes in the specifications and documentation clarifications to the participating members. [\CHG]

Preface

Information contained in this preface addresses how this document is to be used and explains the relationship between this specification and the Open Buying on the Internet™ (OBI) specifications.

Purpose of the Document

The purpose of this document is to provide an implementation guideline for e-business system implementers and solution providers who wish to create interoperable software application components that cooperatively execute RosettaNet "Partner Interface Processes" (PIPs).

Intended Audience

The primary audience for this document is software engineers. These engineers will be developing RosettaNet-compliant networked software applications that can interoperate with other RosettaNet-compliant networked software applications developed by other companies. These applications will cooperatively execute and validate RosettaNet e-business PIP guidelines.

The secondary audience is system architects. This includes both architects within implementing companies who must integrate their architectures with RosettaNet architectures and applications, as well as those who volunteer to participate in RosettaNet projects to create additional RosettaNet e-business specifications.

Prerequisites

It is assumed that the audience will be familiar with or have knowledge of the following:

- OSI layers and protocols
- General Internet protocols
- Digital signatures and SSL
- XML
- All the external references listed in the Bibliography
- Working knowledge of Unified Modeling Language (UML).

Scope of the Document

This document provides sufficient business and architectural background to understand the context for the implementation framework; specification of the RosettaNet implementation framework is the focus of this document. Included with the framework are sections detailing technical compliance, implementation notes, the RosettaNet action/transaction protocol message format, and a complete example of an action/transaction protocol message.

This document does **not** provide user documentation, nor a detailed architectural treatise. This document subsumes previous versions.

Structure of This Document

This document is an implementation guideline for an instance of the RosettaNet networked application architecture. It is divided into the following sections:

1. "Introduction" provides the RosettaNet business model, introduces the concept of "Partner Interface Process" (PIP) guidelines; and discusses the RosettaNet networked application architecture.
2. "Partner Interface Process Specifications" provides the preamble header, service header, and service content message format.
3. "RosettaNet Networked Application Protocols" provides an e-business application framework that uses XML and the WWW to create interoperable software solutions that can execute the RosettaNet service specifications.
4. "Digital Signatures" provides details on digital signatures within the scope of the RosettaNet implementation framework.
5. "Authentication Using SSL and Digital Certificates" provides details on implementing authentication using SSL and digital certificates.
6. "Technical Compliance" comprises technical compliance specifications that ensure interoperability among solutions created by individual supply chain companies and individual RosettaNet solution partners.
7. "Implementation Needs" provides details for implementing some of the technical requirements specified in earlier sections.
8. "RosettaNet Service Protocol Message DTDs" provides a list of the non-PIP-specific message guidelines and DTDs that are necessary for creating RosettaNet-compliant messages. The message guidelines and DTDs themselves are contained in separate individual files.
9. "Complete Example of an Service Protocol Message" uses PIP2A8 to illustrate the service protocol message document instance.

Style Conventions

This specification uses a number of conventions to convey specific meanings. These fall into two categories: typographical conventions and language conventions. They are identified here.

Typographical Conventions

The use of a `monospaced font` indicates that a code fragment is being presented. 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.

Language Conventions

This specification adopts the conventions expressed in the IETF's 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 this document are to be interpreted as described in RFC 2119.

Related Documents

This specification draws heavily upon the work of the Open Buying Initiative (OBI™) and its specifications. Other RosettaNet documents are also heavily drawn upon. Specific documents are listed in the Bibliography at the end of this document.

Acknowledgements

Large sections of the *RosettaNet Implementation Framework Specification* document have been adapted from the *Open Buying on the Internet (OBI) ™ Technical Specifications* (v1.1) document. Paragraphs and sometimes entire sections were copied and modified to suit the more generic needs of RosettaNet.

As an example, OBI's data formats and process descriptions are specific to a catalog service and an ordering process; RosettaNet needs a description of OBI-specified processes and data formats that are applicable to any agent software component and service software component.

To meet RosettaNet's needs, this specification:

1. specifies the "RosettaNet Object" to accommodate upper level protocols as opposed to OBI's EDI-formatted messages,
2. generalizes the "catalog service" to a "business service" and a "requester" to a "user,"
3. generalizes buying and selling organizations, and
4. describes a layered conceptual model with implemented protocols

RosettaNet also references OBI's documented specifications for:

1. the security layer,
2. the transfer layer and
3. the agent layer.

Grateful acknowledgement is made of RosettaNet's reliance upon the previous work of the OBI consortium, particularly in Section 3 of this RosettaNet document. The following words or phrases were routinely substituted in the OBI content to make it suitable for RosettaNet needs.

Table 1. RosettaNet Substitution of OBI Terminology

OBI	RosettaNet	Definition
OBI	RosettaNet	The RosettaNet Consortium
Buying Organization	Initiating Organization	An organization that initiates a RosettaNet service request.
Selling Organization	Servicing Organization	An organization that services a RosettaNet service request.
Requisitioner	User	A person that requests RosettaNet services via a web browser.
Catalog	Service	A business service.

1 INTRODUCTION

RosettaNet's mission is to harness the global and pervasive reach of the Internet by defining, and leading the implementation of, open and common electronic business processes. These processes are designed to align the electronic business interfaces between participating supply chain partners, resulting in measurable benefits. Current partners include those in the Information Technology (IT) Products supply chain and those in the Electronic Component (EC) supply chain.

RosettaNet calls these open and common e-business processes "Partner Interface Process" (PIP) guidelines. These PIP guidelines comprise both human-readable and machine-readable specifications that are used to monitor the compliance of partner e-business implementations and applications to the RosettaNet specifications. RosettaNet PIP guideline development follows a methodology that is outlined in this section. PIP guidelines are distributed to partners in the supply chain once they have been agreed upon.

Partner organizations in the member supply chains will use PIP guidelines to configure their specific e-business processes with other partner organizations. An important component of these e-business processes are the networked computer applications that collaboratively execute e-business processes that span partner organizations in the supply chain. These applications inter-operate by adhering to the open and common RosettaNet networked-application architecture.

This document provides an implementation guideline for a specific networked application that is an instance of the RosettaNet architecture. Applications built to this specification are intended to operate in unison with the RosettaNet business model in which partner interface process specifications are developed and distributed to partners in the participating supply chains. Part of the business model requires applications that adhere to the RosettaNet networked application architecture to be configured to collaboratively execute these partner interface processes as part of their overall e-business processes.

1.1 ROSETTANET BUSINESS MODEL

Figure 1 illustrates the conceptual RosettaNet business model. The RosettaNet model is intended to enable supply chain business partners to execute interoperable e-business processes by continuously developing, maintaining and distributing partner interface process implementation guidelines.

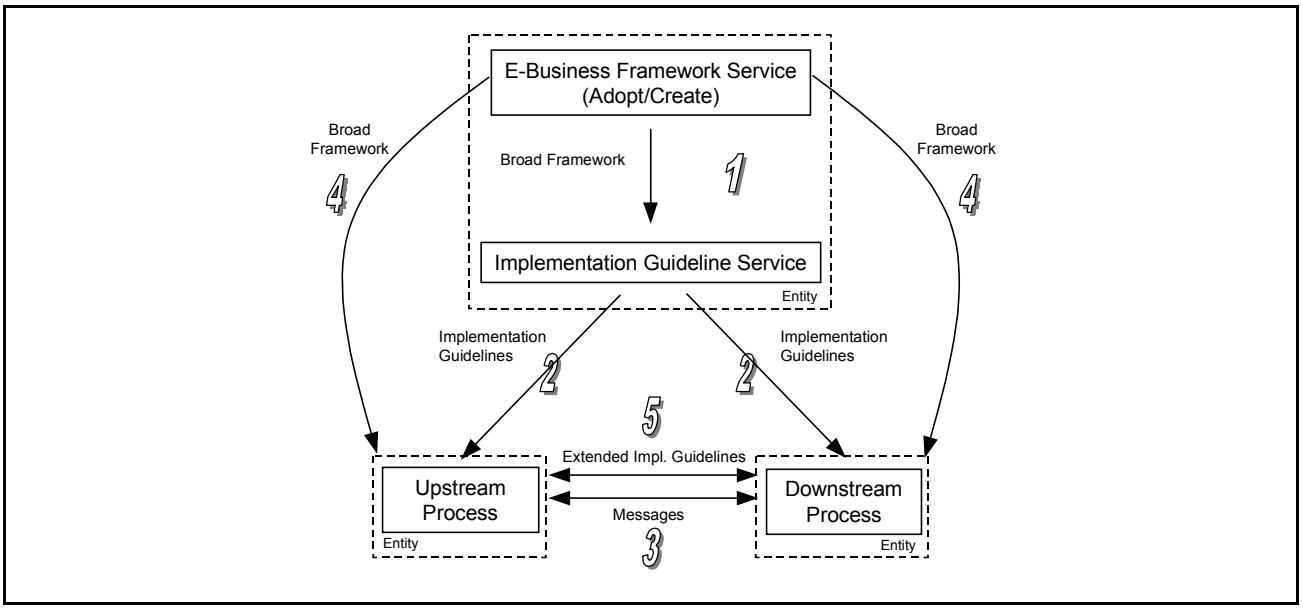


Figure 1. RosettaNet e-Business Model

RosettaNet adopts existing e-business standards, guidelines, or specifications wherever possible and creates new e-business framework specifications where necessary. Typically these frameworks are generic and all-embracing in nature so that they can be used for all types of e-business applications. There are five conceptual parts to the RosettaNet business model:

1. RosettaNet's Partner Interface Process (PIP) teams use these frameworks to create PIP guidelines (labeled "1" in Figure 1) that define how computer systems will cooperatively execute e-business processes in the supply chain. These guidelines narrow the general information frameworks into detailed specifications that must be embraced by all members who wish to conduct e-business with RosettaNet-compliant partners.
2. The implementation guidelines are provided to companies who wish to conduct e-business according to the RosettaNet's specifications (labeled "2" in Figure 1).
3. Guidelines are used to validate the information exchanged between companies (labeled "3" in Figure 1). These guidelines can also be used to create the content that is exchanged and to support tools used to create and manage content in each company's internal system.
4. RosettaNet intends to allow companies to extend the implementation guideline for their own individual needs. Companies can extend the implementation guideline according to the broad framework (labeled "4" in Figure 1). These extensions cannot override those specified by RosettaNet.
5. The extended implementation guidelines are then exchanged between companies (labeled "5" in Figure 1). This then allows companies to validate these message extensions during exchange.

A unique aspect of the RosettaNet e-business model is that all guidelines and translations will be distributed as machine-readable documents. This will allow companies to quickly configure their RosettaNet-compliant applications to execute and validate new or updated PIP specifications.

1.2 Partner Interface Process (PIP) Guidelines

RosettaNet has evolved a methodology for developing PIP guidelines as shown in Table 2. Companies that are part of the member supply chains collaboratively develop these PIP guidelines in three steps[1]. These companies and RosettaNet solution partners then implement the guidelines as interoperable, networked software systems.

Table 2. RosettaNet's PIP Development Methodology

Step	Process Model	Domain	Interaction	Exchange
1	"as-is" Business Process	Organization with Suppliers	Employee Role Interaction/ External Process Interaction	Business Information in any format
2	Partner Interface Process	Partner Roles in Supply Chain	Partner Role Interaction	Electronic Business Properties
3	Networked Application Execution Process	Software Agents and Services in Communications Network	Service Interaction	Messages

Three steps are necessary in order to develop a PIP guideline, as follows:

1. The first step in the methodology results in a process model that captures the current "as-is" business processes from the perspective of an organization with suppliers. The purpose of this model is to create an understanding of the current partner business processes, their employee role interactions and the business information that they exchange using any number of formats and methods, e.g., telephone, fax, e-mail, WWW.
2. The next step in the methodology uses the current business process model to create a "to-be" business process model for partner types in the supply chain, i.e., Manufacturer, Distributor, Retailer, Financier, Carrier and End User[2]. This "to-be" model comprises partner roles and their partner interface processes. Partner interface processes show the interaction between partner roles that are independent of any organizational configuration. The purpose of this model is to define partner roles, e.g., Catalog Publisher, Catalog Distributor, Order Manager, Requisition Manager, and the structured properties that they exchange when they interact. The defined properties are captured in two property databases — a technical specification property database and a business property database.
3. The e-business process model between partner roles is then used as functional requirements for a networked application execution process model, called a "PIP Blueprint." This model specifies how software agents and services execute partner interface processes in a collaborative networked computer system. Agent and service software components exchange messages in transactions that are in turn sequenced by execution processes. Transactions are units of work where each party in the interaction is either able

[1] There is an additional part to a PIP guideline that provides metrics comparing a partner type's current business process to future e-business process based on RosettaNet specifications, which is not included here. Refer to the RosettaNet white paper entitled *Partner Interface Process Technical Architecture.*

[2] Note that an end user is an organization such as the government. This is not the same entity as an individual consumer.

to commit that the work was performed or must rollback to a state before the transaction was initiated. Execution processes comprise conditional processing that is able to choreograph transactions based on their reported success or failure.

4. Finally, the PIP "blueprints" are used to create a PIP specification, which includes specific business message guidelines to carry out the business processes contained in the PIP and corresponding DTDs for each message guideline.

Networked applications that implement RosettaNet partner interface processes are required to comply with three specifications that result from the final step in the PIP development methodology. These specifications are:

1. *Action specifications*. These specifications comprise the business action messages that are exchanged in a sequence by software agent and service components.
2. *Transaction specifications*. These specifications comprise the sequence of message exchanges that comprise a unit of work. I.e., all parties agree to commit to do the work or they all roll back to a state before the transaction was initiated.
3. *Process specifications*. These execution process specifications conditionally choreograph transactions required for executing a partner interface process.

Supply chain companies and RosettaNet solution partners that wish to create open, interoperable, networked applications need to adhere to these specifications, which are distributed in both human-readable and machine-readable forms. However, the machine-readable versions (i.e., XML DTDs) are not complete specifications, due to the limitations of DTDs themselves. Hence the complete specifications only exist in the human-readable PIP specification and accompanying message guidelines (HTML format). In the future, these specifications will be increasingly machine-readable, allowing RosettaNet member companies to create solutions that can be rapidly configured to changing supply chain business models.

1.3 RosettaNet Networked Application Architecture

RosettaNet's networked application architecture takes as its starting point the International Standards Organization (ISO) Open Systems Interconnect (OSI) reference model shown in Figure 2. The ISO/OSI model defines seven layers: data link, logical link, network, transport, session, presentation and application. These layers can be categorized as providing either transport functionality or application functionality.

The RosettaNet e-business communications model specifically defines the behavior that should occur within the OSI application and session layers. The RosettaNet model divides the OSI application layer into the following sub-layers:

1. *Action Layer*. This layer provides business actions that act either on or with accompanying information.
2. *Transaction Layer*. This layer provides transaction monitoring for sequences of message exchanges that perform a unit of work. Either all parties to the transaction commit to the unit of work of they all roll back to a previous state before the transaction was started.
3. *Process Layer*. This layer encapsulates conditional choreography of transactions for executing a partner interface process.

4. *Service Layer*. This layer provides network resources that perform network and business related functions.
5. *Agent Layer*. This layer provides communication interfaces for user and machine agents.
6. *Message Handling Layer*. This layer provides reliable, asynchronous and scaleable information delivery.
7. *Transfer Layer*. This layer provides information transfer between uniquely named network resources.
8. *Security Layer*. This layer provides a secure communications channel (connection) that, together with digital signatures, can be used to implement authorization and authentication.

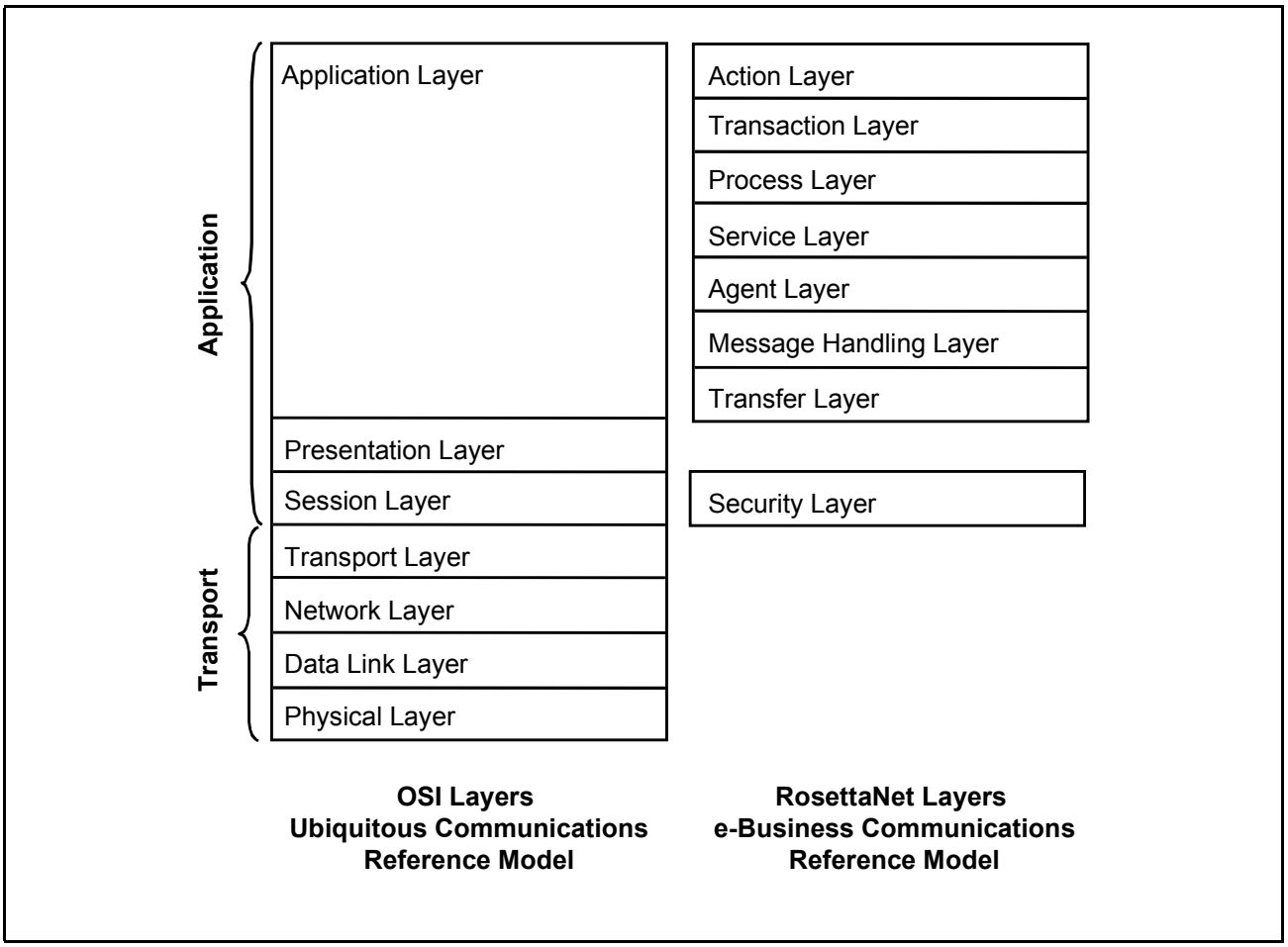


Figure 2. ISO/OSI and RosettaNet Communications Reference Models

The RosettaNet e-business model is a conceptual model. Figure 3 shows the "blueprint" architecture for specifying one or more protocols at each layer of the model to enable RosettaNet networked applications to execute and validate PIP specifications.

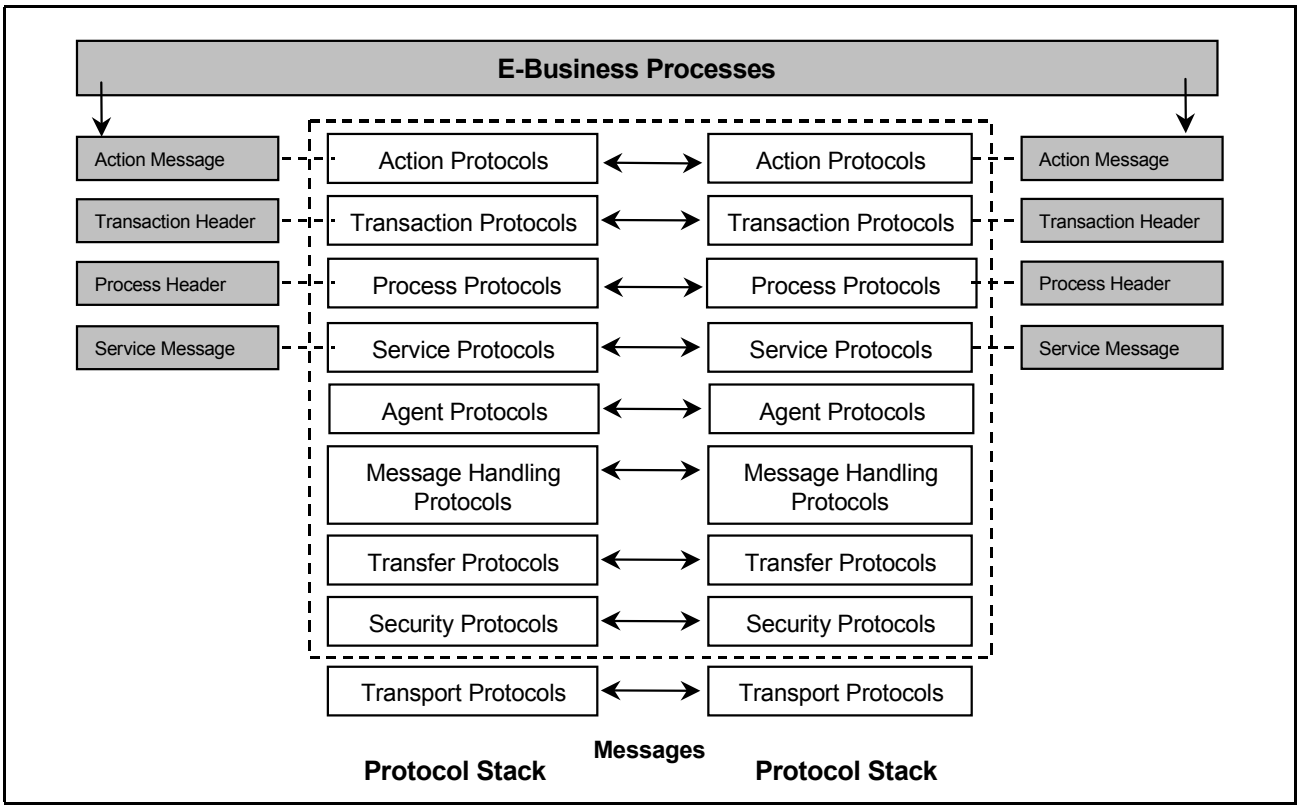


Figure 3. Implementation Framework Conceptual View

The shaded components and the dashed line in Figure 3 are of great significance. RosettaNet is an industry consortium and is primarily in the business of creating PIP guidelines with which their partner organization members agree to conduct e-business. This is shown in the shaded areas. It is expected that RosettaNet and members of the industry will implement these guidelines in a manner that is compatible with their existing information systems and their future information system plans. It is also expected that industry members will build application systems that can accommodate both RosettaNet guidelines and other existing industry guidelines that are used in commercial applications that are outside RosettaNet's scope. The intent of the protocol stack inside the dashed box is to provide an implementation framework for these two benefits to be realized.

2 Partner Interface Process (PIP) Specifications

Fundamental to PIPs is the exchange of business data between partners in a supply chain. RosettaNet-compliant, networked applications receive data in a standard format that they can easily process for their respective line of business functions and applications.

Currently, RosettaNet PIP teams create RosettaNet messages using the set of elements and codes defined in the Rosetta business and technical dictionaries. This set is used as a baseline from which PIP teams create specific message exchange specifications. PIP teams precisely define the values and codes that can be assigned to each of the data elements. This specification set is known as a PIP implementation guideline.

These guidelines are sent from RosettaNet to supply chain partners and RosettaNet solution partners in a human readable printed form, as well as in machine-readable form. The guidelines define the vocabulary, structure and allowable data element values and value types, for each message exchanged during the execution of a PIP. RosettaNet-compliant networked applications must author and read these messages according to the RosettaNet message guideline. These guidelines can additionally be used for validating the messages received in PIP interchanges by translation systems.

PIP specifications enable the development of interoperatable applications. There are three parts to a PIP business message guideline: the Preamble Header, the Service Header, and Service Content. These are all packaged for transport as MIME messages. Each of these are described below.

2.1 PIP Business Message Structure

RosettaNet business messages consist of a message header and a message body. Both the header and the body are complete, valid XML documents. The header and the body are encoded within a multipart/Related MIME message. This section outlines the use of MIME and specifically the "Multipart/Related" content-type to pack different pieces of a RosettaNet business message.

The RosettaNet business message is transported as a MIME message with XML body-parts, as specified below:

```
MIME-Version: 1.0
     Content-Type: Multipart/Related; boundary="RN-part-boundary";
                           type="Application/x-RosettaNet"
     Content-Description: This is the RosettaNet business message

     --RN-part-boundary
     Content-Type:  Application/XML; RNSubType="preamble-header"
     Content-Description: This is the Preamble Header part of the business message

     [The PREAMBLE HEADER goes here]

     --RN-part-boundary
     Content-Type:  Application/XML; RNSubType="service-header"
     Content-Description: This is the Service Header part of the business message

     [The SERVICE HEADER goes here]

     --RN-part-boundary
     Content-Type:  Application/XML; RNSubType="service-content"
     Content-Description: This is the Service Content part of the business message

     [The SERVICE CONTENT goes here]

     --RN-part-boundary--
```

See section 9 for a sample document.

2.1.1 Message Preamble

The preamble section of the MIME message contains elements that are global to the RosettaNet service and those that are common to the Service Header and Service Content

(see below). It is specified with a DTD that is common across all messages. See the *Preamble Guideline* and the *Preamble DTD* for complete documentation.

2.1.2 Message Header

The message header is specified with a DTD that is common across all messages. A separate DTD and/or XML schema for each message will validate the body of the messages. While a single DTD is required to validate all message headers, we expect each message to have its own DTD and/or XML schema.

The rationale behind a common message header DTD that is separate from the message content DTD is primarily because it supports a logical segmentation of validation steps. RosettaNet specifies that incoming messages go through the following validation steps:

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

Grammar and sequence validations are performed against the message header DTD. Validation of the message content is deferred until the schema validation step. Note that the message header may be valid against its DTD, even though the message content may contain errors. The ability to parse the header in such cases separate from the body allows the recipient to retrieve information about the sending party in order to send a failure message.

2.1.3 Message Content

Message content is specified in individual PIPs. Each PIP has one or more "actions" that are described by means of individual DTDs or schema.

2.2 RosettaNet Message Guideline Format

Specification of message guidelines is in human-readable form, using RTF and HTML formats. Additionally, message guidelines are provided in machine-readable formats. The preferred format is XML DTDs. (As various XML schema languages mature and members express the business message guidelines in them (as well as in DTDs). It is expected that W3C's XML Schema (when available) will be among the first.) Message vocabulary comes from RosettaNet dictionaries; each message guideline has its own DTD or schema.

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

DTDs are well understood and there are plenty of parsing tools available to validate the message structures. However, DTDs alone are not sufficient to validate a message at a higher level, such as semantics that may include constraints (absence, presence, etc.) on the elements of a message structure. Unfortunately, there are no mature and open mechanisms for specifying these constraints with commerical off-the-shelf (COTS) tools available today. (Note that schema validation tools will be able to validate more of the message than DTD validation tools.)

Supply chain partners should review their trading partner agreements in this respect. The UN/EDIFACT and American Legal Association recommend that partners agree on the point at which a message is legally considered "received" i.e. the point at which you could send back an acknowledgement of receipt. Such agreement must take into account what partners can do with tools and must be human-validated at this point. RosettaNet is separately working on recommendations for member Trading Partner Agreements.

3 RosettaNet Networked Application Protocols

Figure 4 shows how the Internet and the WWW implement the ISO/OSI layers. The TCP/IP protocol provides the functionality defined for the OSI network and transport layers. Sockets (not shown) and the Secure Sockets Layer (SSL) are OSI session layer protocols. Note that SSL actually comprises a number of sub-layers and protocols but this detail in not shown in the diagram. Figure 4 also shows the two HTTP protocols that operate on top of TCP/IP. With HTTPS, SSL is utilized, and with HTTP, SSL is not utilized.

The relationship of the RosettaNet e-Business application model (as presented in this document) to the OSI layers is also shown in Figure 4. The RosettaNet model specifies that either HTTP or HTTPS (HTTP over SSL) can be used for the transfer and security layers.

The OSI presentation layer is not shown, as neither Internet applications nor RosettaNet networked applications use any protocols from that layer.

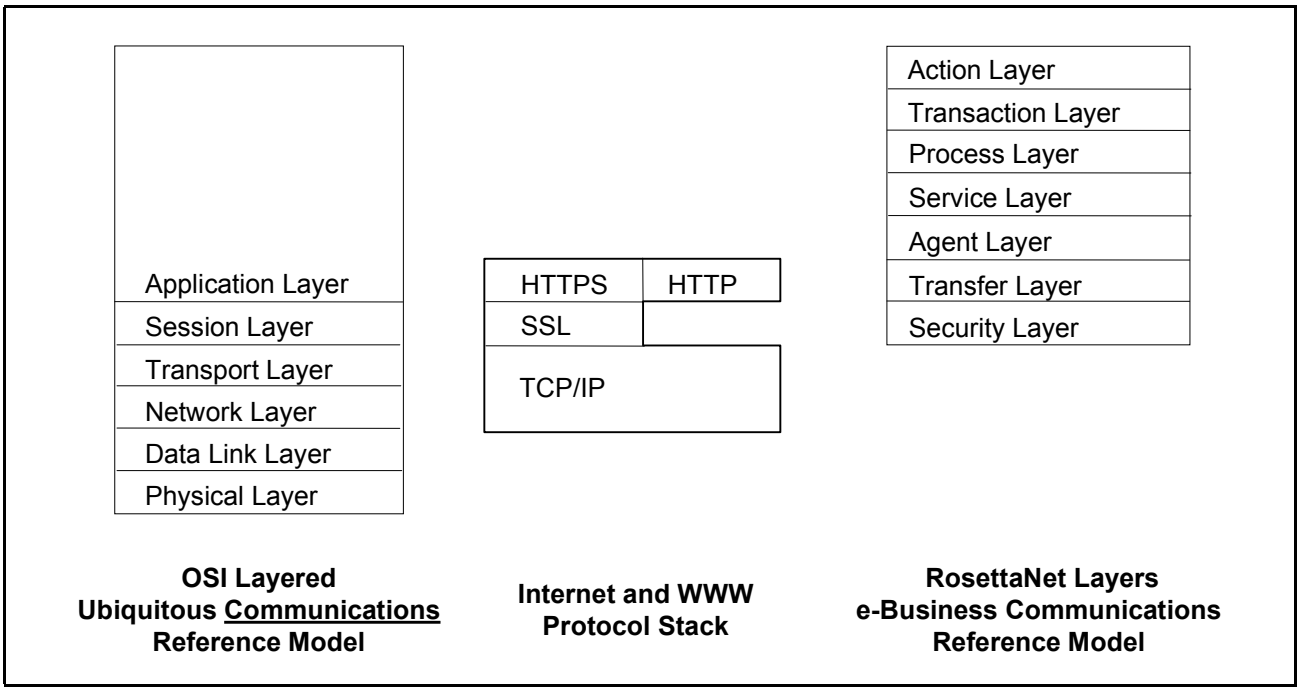


Figure 4. Adapting and Extending the Internet and WWW Protocols

Figure 5 shows the protocols that comprise the RosettaNet implementation framework.

Note that RosettaNet has combined the Service, Process, Transaction, and Action layers from the Implementation Framework conceptual view (Figure 2) into a single Service message for the purposes of the current implementation framework. For agent, service, process, transaction and action layers, the RosettaNet model makes use of several protocols defined by official standards bodies and industry consortiums and organizations.

The agent layer comprises a service-agent protocol, a user-agent protocol and a CGI (Common Gateway Interface) protocol. The RosettaNet Service protocol, with an XML interchange format, is specified for the service, process, transaction and action layers.

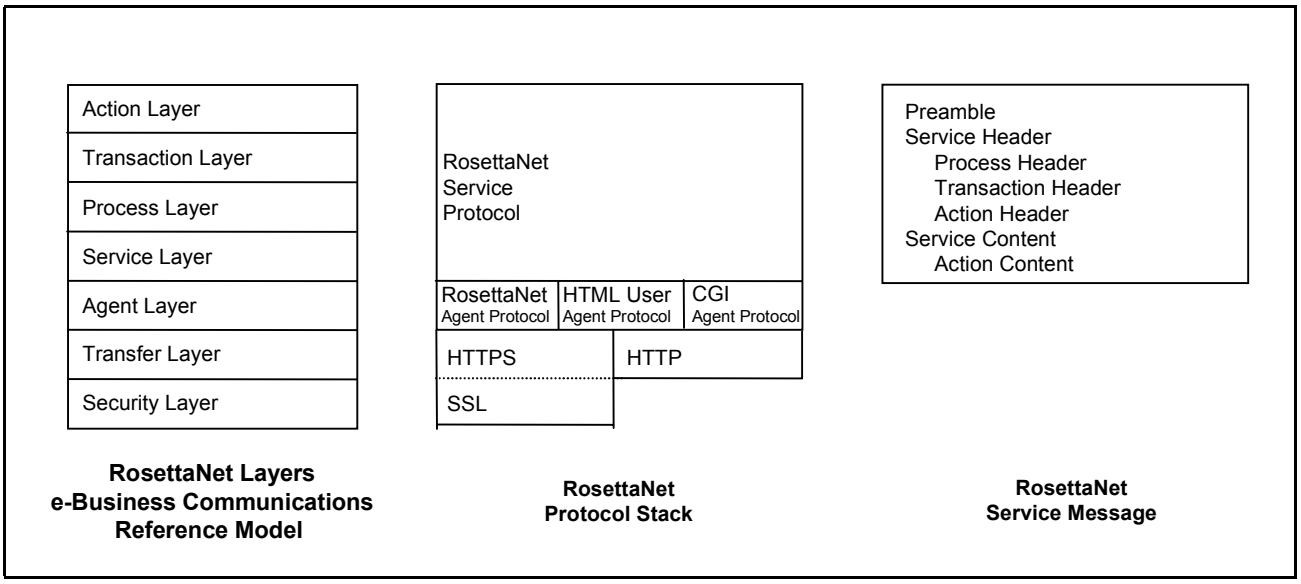


Figure 5. RosettaNet Protocol Stack

A message may comprise a header, content and a trailer as shown in Figure 6. The content may be a message from an upper protocol or it may be application information. Note that there is no trailer for the service protocol message.

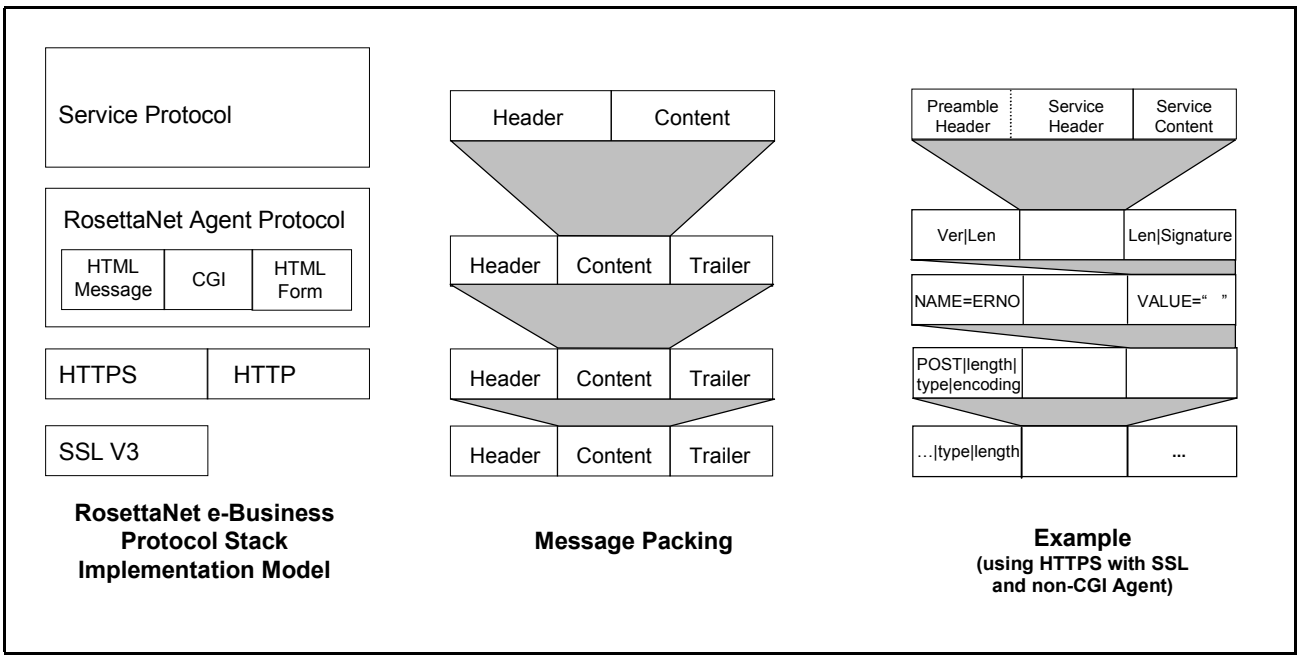


Figure 6. Message Packing Example for the RosettaNet Protocol Stack

The action message is encapsulated into a service message. This message is in turn encapsulated within a header and trailer specified by the agent protocol. The agent protocol facilitates communication between applications that support the RosettaNet agent protocol.

The resulting RosettaNet "object" can then be directly encapsulated into an HTTP message, into an HTML form, or into a CGI name-value pair. Example 1 shows the RosettaNet Object being encapsulated into an HTML form and the form itself being encapsulated in an HTTP message. From HTTP downwards, the messages are handled by the lower layers like any other messages. The process continues down the stack until the physical layer where it is sent over a communications medium for unpacking by a receiving computing system.

3.1 Message-Packing Example

The "Distribute Stock Keeping Unit Creation Notification" RosettaNet message in *PIP 2A8: Distribute Product Stock Keeping Unit* that is sent from a "Buyer" to a "Customer Manger" is used in the following message-packing example.

In this example, the message exchange is between two RosettaNet services, namely, the "Buyer Service" and the "Customer Manager Service". This is a service-to-service exchange and in this case, the RosettaNet Object is directly encapsulated by the HTTP protocol at the transport level. In general, agents can also act on behalf of users (as in the case of a web browser). When agents act on behalf of users, the RosettaNet Object is encapsulated into an HTML form as a CGI name-value pair, which in turn is exchanged via the HTTP protocol.

Note that the example ends at the HTTP protocol but in a working implementation the message is encapsulated in a TCP message that is in turn encapsulated in an IP message and so on down the protocol stack.

The following sections describe the various parts of the business message. Referring to the example given in this section will assist in understanding the below explanations.

3.1.1 RosettaNet Service Protocol Message

The RosettaNet Service protocol message is the business message that is exchanged between two RosettaNet entities (Services and/or Agents). As described earlier, the service protocol message comprises a "Preamble", "Service Header" and the "Service Content" that are packaged into a MIME multipart/related content-type.

The Preamble, Service Header and the Service Content are all XML documents which need to be validated against their RosettaNet-provided DTDs. The Preamble and the Service Header have DTDs that are common to all RosettaNet Service Messages and are defined and explained by RosettaNet's *Preamble Part Message Guideline* and *Service Header Part Message Guideline* respectively. The Service Header itself comprises elements for the Process Header, Transaction Header and the Action Header. The Service Content varies based on the actual business message being exchanged (as defined in individual PIPs) with each business message that can be a Service Content defined by a RosettaNet guideline of its own.

Below we look at each of the above components in some detail and how they are packaged together into a Service Message.

3.1.1.1 Preamble

The Preamble is the first component of the service message and contains elements that are global to the RosettaNet service and the elements that are common to the Service Header and the Service Content.

In Figure 7, an example instance of the preamble is shown. All elements of the preamble and example values for the elements are shown. Note that the actual element tag names are correct as of this writing; however, the actual *Preamble Part Message Guideline* published by RosettaNet should be consulted for the correct tag names for the elements, the set of valid values, the data-type and other semantics of the element values etc. This also applies to all other components of the service message (i.e. Service Header and Service Content).

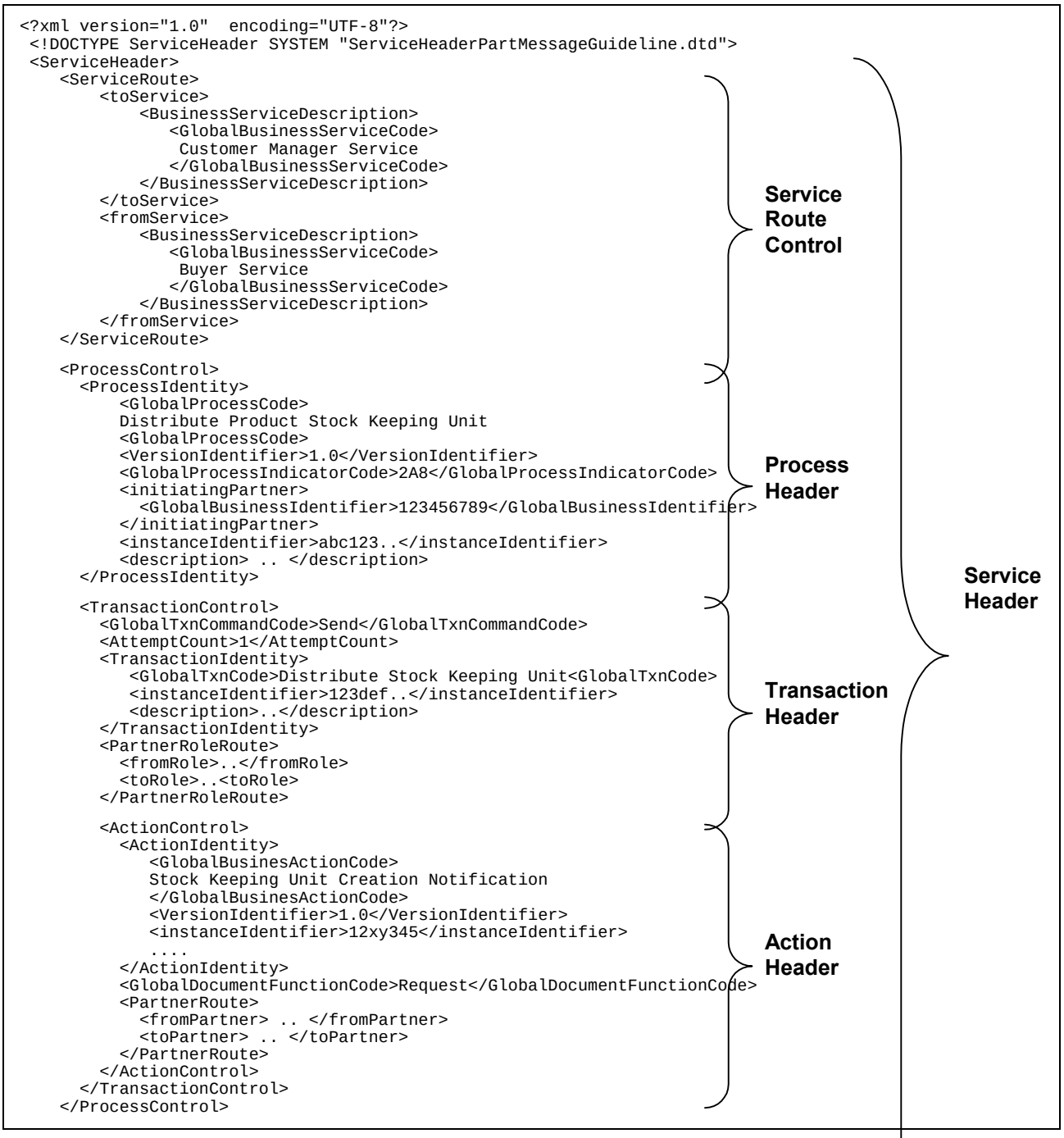
```
<?xml version="1.0"  encoding="UTF-8"?>
<!DOCTYPE Preamble System "PreamblePartMessageGuideline.dtd">
<Preamble>
   <VersionIdentifier>1.1</VersionIdentifier>
   <DateTimeStamp>19990531T132000.0500Z</DateTimeStamp>
   <GlobalAdministeringAuthorityCode>RosettaNet</GlobalAdministeringAuthorityCode>
   <GlobalUsageCode>Test</GlobalUsageCode>
</Preamble>
```

Figure 7. Preamble Header

3.1.1.2 Service Header

In Figure 8, an example instance of the service header is shown. Not all elements and their specific values are shown for conciseness. Refer to the service header guideline published by RosettaNet for the correct tag names for the elements, the set of valid values, the data-type and other semantics of the element values etc.

As shown in the diagram the service header contains sub-elements for the Service Route, Process, Transaction and Action Header parts. The Service Route identifies the sender and recipient services of the business message. Process header contains elements that identify the RosettaNet process, version, global process code, process instance identifier etc. The Transaction Header contains information relating to the RosettaNet Transaction like gobal transaction code, transaction instance identifier etc. The Action Header contains information relating to the RosettaNet Business Action message that forms the Content of the Service Message. This information includes the Global Business Action Identifier, version, instance identifier, sending and receiving partner global business identifier etc.




```
<?xml version="1.0"  encoding="UTF-8"?>
 <!DOCTYPE ServiceHeader SYSTEM "ServiceHeaderPartMessageGuideline.dtd">
 <ServiceHeader>
    <ServiceRoute>
        <toService>
            <BusinessServiceDescription>
               <GlobalBusinessServiceCode>
                Customer Manager Service
               </GlobalBusinessServiceCode>
            </BusinessServiceDescription>
        </toService>
        <fromService>
            <BusinessServiceDescription>
               <GlobalBusinessServiceCode>
                Buyer Service
               </GlobalBusinessServiceCode>
            </BusinessServiceDescription>
        </fromService>
    </ServiceRoute>

    <ProcessControl>
      <ProcessIdentity>
          <GlobalProcessCode>
          Distribute Product Stock Keeping Unit
          <GlobalProcessCode>
          <VersionIdentifier>1.0</VersionIdentifier>
          <GlobalProcessIndicatorCode>2A8</GlobalProcessIndicatorCode>
          <initiatingPartner>
            <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
          </initiatingPartner>
          <instanceIdentifier>abc123..</instanceIdentifier>
          <description> .. </description>
      </ProcessIdentity>

      <TransactionControl>
        <GlobalTxnCommandCode>Send</GlobalTxnCommandCode>
        <AttemptCount>1</AttemptCount>
        <TransactionIdentity>
           <GlobalTxnCode>Distribute Stock Keeping Unit<GlobalTxnCode>
           <instanceIdentifier>123def..</instanceIdentifier>
           <description>..</description>
        </TransactionIdentity>
        <PartnerRoleRoute>
          <fromRole>..</fromRole>
          <toRole>..<toRole>
        </PartnerRoleRoute>

        <ActionControl>
          <ActionIdentity>
             <GlobalBusinesActionCode>
             Stock Keeping Unit Creation Notification
             </GlobalBusinesActionCode>
             <VersionIdentifier>1.0</VersionIdentifier>
             <instanceIdentifier>12xy345</instanceIdentifier>
             ....
          </ActionIdentity>
          <GlobalDocumentFunctionCode>Request</GlobalDocumentFunctionCode>
          <PartnerRoute>
            <fromPartner> .. </fromPartner>
            <toPartner> .. </toPartner>
          </PartnerRoute>
        </ActionControl>
      </TransactionControl>
    </ProcessControl>
```

Figure 8. Service Header Instance

3.1.1.3 Service Content

As described earlier, the service content is the RosettaNet Business Action message. This is the fundamental business message that is exchanged within in a transaction of a RosettaNet Business Process and is specified in corresponding PIPs.

In Figure 9, an example instance of the service header is shown. Again, not all elements and their specific values are shown for conciseness. For the specific business message of interest, refer to the relevant individual action message guideline published by RosettaNet for the correct element names, the set of valid values, the data-type and other semantics of the element values etc. The action message used in this figure is for the "Stock Keeping Unit Creation Notification" business action.

```
<?xml version="1.0"  encoding="UTF-8"?>
<!DOCTYPE Pip2A8ProductNotification SYSTEM "PIP2A8SKUCreationNotificationGuideline.dtd">
<PIP2A8SKUCreationNotification>
<ProductNotice>
  <theNotice><FreeFormText>Produt SKU created.</FreeFormText></theNotice>
  <GlobalProductIdentifier>00123456789012</GlobalProductIdentifier>
</ProductNotice>
<fromRole>
  <PartnerRoleDescription>
    <GlobalPartnerRoleClassificationCode>Buyer</GlobalPartnerRoleClassificationCode>
    <PartnerDescription>
     <GlobalPartnerClassificationCode>Distributor</GlobalPartnerClassificationCode>
     <BusinessDescription>
       <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>
       ...
     </BusinessDescription>
    </PartnerDescription>
    <ContactInformation>
     <contactName><FreeFormText>A. Name</FreeFormText></contactName>
     <telephoneNumber><CommunicationsNumber>299.688.8998</CommunicationsNumber></telephoneNumber>
     <EmailAddress>email@mail.com</EmailAddress
    </ContactInformation>
  </PartnerRoleDescription>
</fromRole>
<toRole>
  <PartnerRoleDescription>
    <GlobalPartnerRoleClassificationCode>Customer Manager</GlobalPartnerRoleClassificationCode>
    <PartnerDescription>
     <GlobalPartnerClassificationCode>Manufacturer</GlobalPartnerClassificationCode>
     <BusinessDescription>
       <GlobalBusinessIdentifier>987654321</GlobalBusinessIdentifier>
       ...
     </BusinessDescription>
    </PartnerDescription>
    <ContactInformation>
     <contactName><FreeFormText>B. Name</FreeFormText></contactName>
     <telephoneNumber><CommunicationsNumber>188.688.8998</CommunicationsNumber></telephoneNumber>
     <EmailAddress>abc@xyz.com</EmailAddress
    </ContactInformation>
  </PartnerRoleDescription>
</toRole>
<thisDocumentGenerationDateTime>
   <DateTimeStamp>19990530T132000.0500Z</DateTimeStamp>
</thisDocumentGenerationDateTime>
<thisDocumentIdentifier>
 <ProprietaryDocumentIdentifier>DOC-001</ProprietaryDocumentIdentifier>
</thisDocumentIdentifier>
</PIP2A8SKUCreationNotification>
```

Figure 9. Action Message Instance

The Preamble, Service Header and Service Content are packaged into MIME multipart/related Service Message.

In Figure 10, an example MIME-packaged service message is shown with the respective XML parts for preamble, service header and content. The process, transaction and action header sub-elements of the service header are also shown. However, not all elements and their specific

values have been shown for conciseness. A complete example of a Service Protocol Message is shown in section 9 of this document.

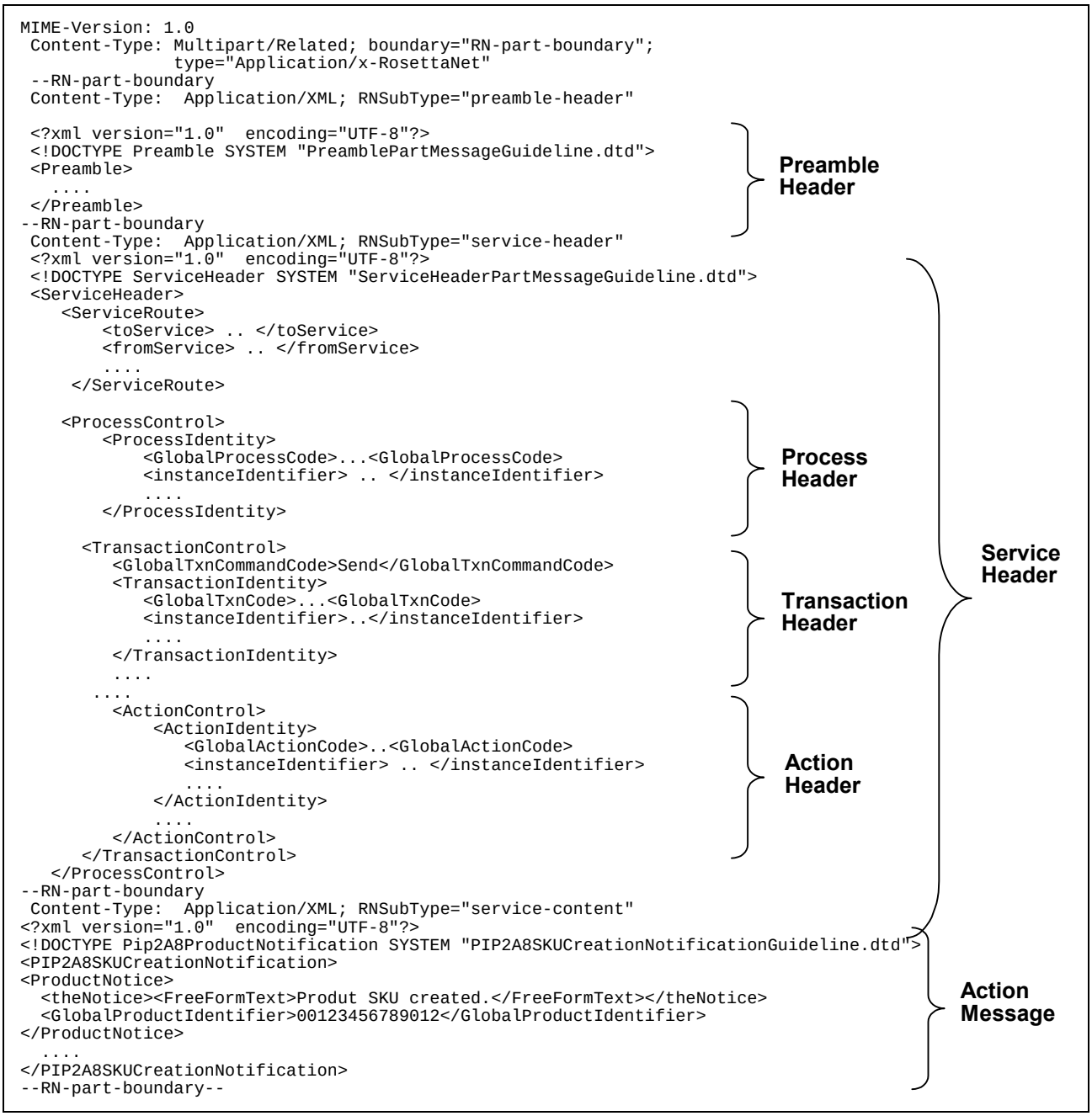


Figure 10. MIME-Packaged Service Protocol Message Instance

3.1.2 RosettaNet Agent Protocol Message

The service protocol message is encapsulated into a RosettaNet agent protocol message (object transferred at the Agent Layer) termed the "RosettaNet Object." Note that the structure of this object is equivalent to that specified in the *OBI Technical Specification*, v.1.1.

The RosettaNet Object is composed of a version and content length header, content comprising a message from an upper protocol, and a digital signature length followed by a digital signature trailer. Figure 11 shows an example of the RosettaNet agent protocol message that encapsulates the service protocol message.

The RN version field is four bytes (big endian format) in length, with an implicit period after two bytes. To represent version 1.10, this field in hexadecimal format would be 0x00010100. Software reading these four bytes buffer would split it into two numbers, each of size two bytes, and interpret them as major and minor versions.

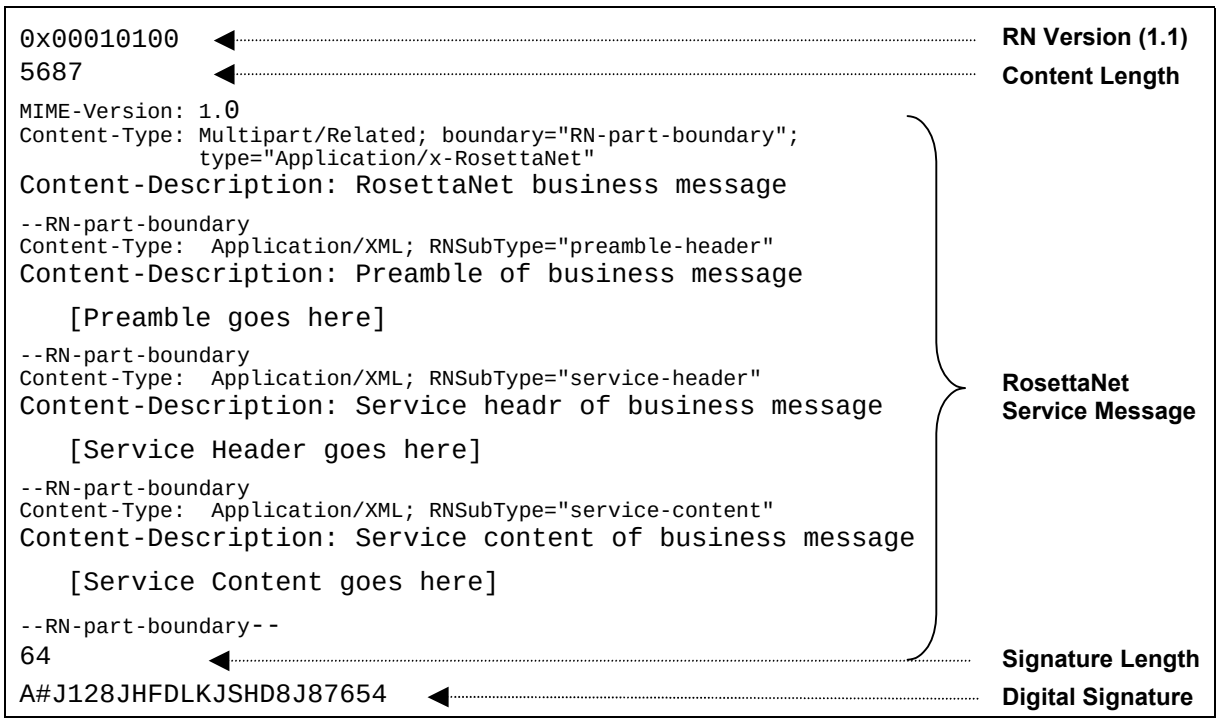


Figure 11. RosettaNet Object Format

The message packing technique used for this message follows that typically used for communications protocol messages. All message parts are arranged in a sequential manner and the size of fixed fields precedes variable length fields.

3.1.2.1 HTML User Agent Protocol Message

This section outlines a user agent protocol for communications with applications such as web browsers. It is included for completeness although it is not part of the previous example showing message exchange between RosettaNet services. This message protocol facilitates transmission of upper protocol messages (RosettaNet Objects) from services to a user agent

such as a web browser. The vocabulary, structure and interchange format for this protocol's message is taken from that specified for an HTML form. RosettaNet Objects are encapsulated into the value field for a hidden input element in an HTML form. Before the object is actually placed into the value field for the VALUE attribute it is encoded in base64. This method of encapsulation is shown as follows:

```
<INPUT TYPE=HIDDEN NAME=ERNO VALUE="the_base64_encoded_RosettaNet_Object">
```

This user agent protocol message is packed as an HTML document comprising a "Form." Example 1 is markup showing an HTML document instance that can be used for message packing.

Example 1. Agent Protocol Message

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
   <HEAD>
<TITLE>Subscription Request</TITLE>
   </HEAD>
   <BODY>
      <FORM METHOD=POST ACTION="http://www.example.com/rn-agent/submit" >
      <INPUT TYPE=HIDDEN NAME=ERNO
       VALUE="the_base64_encoded_RosettNet_Object">
      <INPUT TYPE=SUBMIT NAME="Submit">
      </FORM>
   </BODY>
</HTML>
```

3.1.2.2 Common Gateway Interface Agent Protocol Message

This section is included for- completeness, although it is not part of this particular example showing message exchange between RosettaNet services. In addition to its agent protocol, RosettaNet specifies an alternate message exchange protocol based on Common Gateway Interface (CGI), enabling interactions for message exchange among applications that do not support the RosettaNet agent protocol. This message protocol facilitates transmission of upper protocol messages from applications such as web browsers to RosettaNet services. The vocabulary, structure and interchange format for this protocol's message is taken from that specified for the CGI standard. RosettaNet Objects are encapsulated into the value field for a name-value pair. This method of encapsulation is shown in the following example.

```
ERNO=RosettaNet_Object
```

3.1.2.3 Transfer Protocol Message

The transport protocol message encapsulates all the agent protocol messages. The format of this message is that of HTTP version 1.0 or higher. Refer to the HTTP documentation (listed in the Bibliography) for more information on the format of HTTP request and response messages.

The following example shows the transfer-protocol request message for exchanging an encapsulated RosettaNet agent protocol message.

Example 2. Transfer Protocol Request Message

```
POST https://www.example.com/rn-agent/submit HTTP/1.0
Content-length:  3492
Content-type:  application/x-rosettanet-agent; version=1.0

RosettaNet_Object
```

Example 3 shows the transfer-protocol response message for exchanging an encapsulated HTML user agent protocol message.

Example 3. Transfer Protocol Response Message

```
HTTP/1.0 OK 200
Content-length:  3492
Content-type:  text/html

<HTML>
   <HEAD>
      <TITLE>Subscription Request</TITLE>
   </HEAD>
   <BODY>
      <FORM METHOD=POST ACTION="http://www.example.com/rn-agent/submit" >
      <INPUT TYPE=HIDDEN NAME=ERNO
          VALUE="the_base64_encoded_RosettaNet_Object">
      <INPUT TYPE=SUBMIT NAME="Submit">
      </FORM>
   </BODY>
</HTML>
```

Example 4 shows the transfer-protocol request message for exchanging an encapsulated CGI agent protocol message.

Example 4. Transfer Protocol Request Message (CGI Agent)

```
POST https://www.example.com/rn-agent/submit HTTP/1.0
Content-length:  3492
Content-type:  application/x-www-form-urlencoded

ERNO=RosettaNet_Object
```

Refer to the HTTP specification for the message packing method used for this protocol.

3.1.3 Protocols below the Transfer Protocol

If the transfer protocol message exchange requires security features such as encryption, authentication or authorization then the message is encapsulated into the message protocols provided by the Secure Sockets Layer (SSL) protocol. Either the transfer protocol message or the SSL protocol message is then encapsulated in a TCP message. This process of message encapsulation and message packing continues all the way down the protocol stack until finally the message is sent over a communications medium to another computing system. Refer to the respective standards for additional information.

3.2 Transferring RosettaNet Objects between Web Servers

The networked application specified in this document is built on the web protocols and thus exchange information with each other using web servers. There are two methods for transferring RosettaNet Objects between web servers: a) the server-to-server method for directly exchanging information between two web servers, and b) the server-browser-server method for indirectly exchanging information between web servers via a web browser.

3.2.1 Server-to-Server Transfer

The message-packing example in the previous section showed how a RosettaNet Object message is fabricated for transfer by the transfer protocols. An application that transfers this RosettaNet Object to a remote web server via a local web server requests the HTTP protocol to transfer the object as content using the HTTP/1.0 POST request to a target URL. The recipient receives the HTTP request and immediately checks the HTTP headers. If the content-type or transfer encoding is improper, or if the content length fails to match the actual length of the entity body, the recipient returns a 400 (BAD REQUEST) response. If the request is accepted for processing by upper layers in the protocol, a 200 (OK) response will be returned immediately as an acknowledgement of message receipt.

If a sender does not receive a response to the request, then the application must retry the POST method until a response is returned. A receiver application must handle duplicate messages. The method of handling duplicates is not specified.

3.2.2 Server-Browser-Server Transfer

An application can also transfer a RosettaNet Object to a remote web server via a local web server and a web browser. In this case the application requests the HTML User Agent Protocol to first wrap the object as the value for a "VALUE" attribute in a hidden field that is part of an HTML form, before transfer by HTTP as in the server-to-server transfer method. The web browser then forwards the RosettaNet Object by requesting the CGI Agent Protocol to first wrap the object as the value of a name-value pair, before transfer by HTTP as in the server-to-server transfer method.

3.3 RosettaNet Protocol Stack Specification

This RosettaNet implementation framework guideline comprises the following protocol stack that is built on top of the TCP/IP protocol layer. RosettaNet does not need to specify TCP/IP, as the lowest level protocol services used by RosettaNet applications is HTTP and HTTP over SSL.

1. **RosettaNet Upper Layer Protocols.** This includes, action, transaction, process and service protocols. The action protocol comprises rules and conventions that govern the exchange of messages that comprise business actions that are part of a larger business process executing in a distributed e-business computing environment. The transaction protocol comprises rules and conventions that govern the exchange of request and

response messages that comprise a sequential, time-dependant commitment to perform units of work characterized by ACID properties.

2. **Agent Protocols**
   a. **RosettaNet Agent Protocol.** This protocol comprises rules and conventions that govern the exchange of request and response messages between RosettaNet services.
   b. **HTML User Agent Protocol.** This protocol comprises rules and conventions that govern the exchange of response messages from RosettaNet services to HTML user agents such as web browsers.
   c. **CGI Agent Protocol.** This protocol comprises rules and conventions that govern the exchange of messages from applications that request services via a common gateway interface to RosettaNet services.
3. **The HyperText Transfer Protocol.** The HTTP protocol comprises rules and conventions that govern the exchange of objects and object-method invocations that occur over the network. The HTTPS protocol performs the same functions as the HTTP protocol in conjunction with the SSL protocols.
4. **The SSL protocols.** These protocols comprise rules and conventions that govern the exchange of messages in a secure environment. SSL provides channel-level security for encryption, authentication and authorization.

3.3.1 Transaction Model

The transaction model that governs the exchange of service protocol messages is a simple asynchronous request and response model. There is no requirement for multiple message exchanges, multiple action messages per transaction exchange or for a more complex transaction chaining or two phase commits. The need for these advanced transactions will surface as more PIPs are created. The sequence of message exchange is specified in the PIP document.

3.3.2 RosettaNet Agent Protocol

The RosettaNet agent protocol is known as a RosettaNet Object as it is modeled on the OBI object specification. Much of the following specification is taken from the OBI specification and modified to meet the needs of RosettaNet.

The RosettaNet Object comprises five fields as shown in Figure 12. Multi-byte values should use network (or big endian) byte order. The *version* field is four bytes in length. It uses a <major>.<minor> numbering scheme to indicate a version of the RosettaNet Object. The major and minor numbers should be treated as separate two 8-bit integers with the major number in the most significant two bytes and the minor number in the third and fourth bytes.

The "version number" field contains the "RosettaNet Object" version of the RosettaNet Object format which would be represented with the bytes: 0x00010100.

The "data length" field is a 32-bit integer that represents the number of bytes in the RosettaNet content field.

The "Content" field is a variable length string containing a message from a protocol higher up the protocol stack (Service Message).

The "signature length" field is a 32-bit integer that represents the length in bytes of the signature field. This field must always be present and should be 0 if no signature is included.

The "signature" field is a variable length field that contains a signature on the contents of the RosettaNet content field. The content of the signature field is a BER-encoded PKCS #7 data object. If no signature is included, this field is empty.

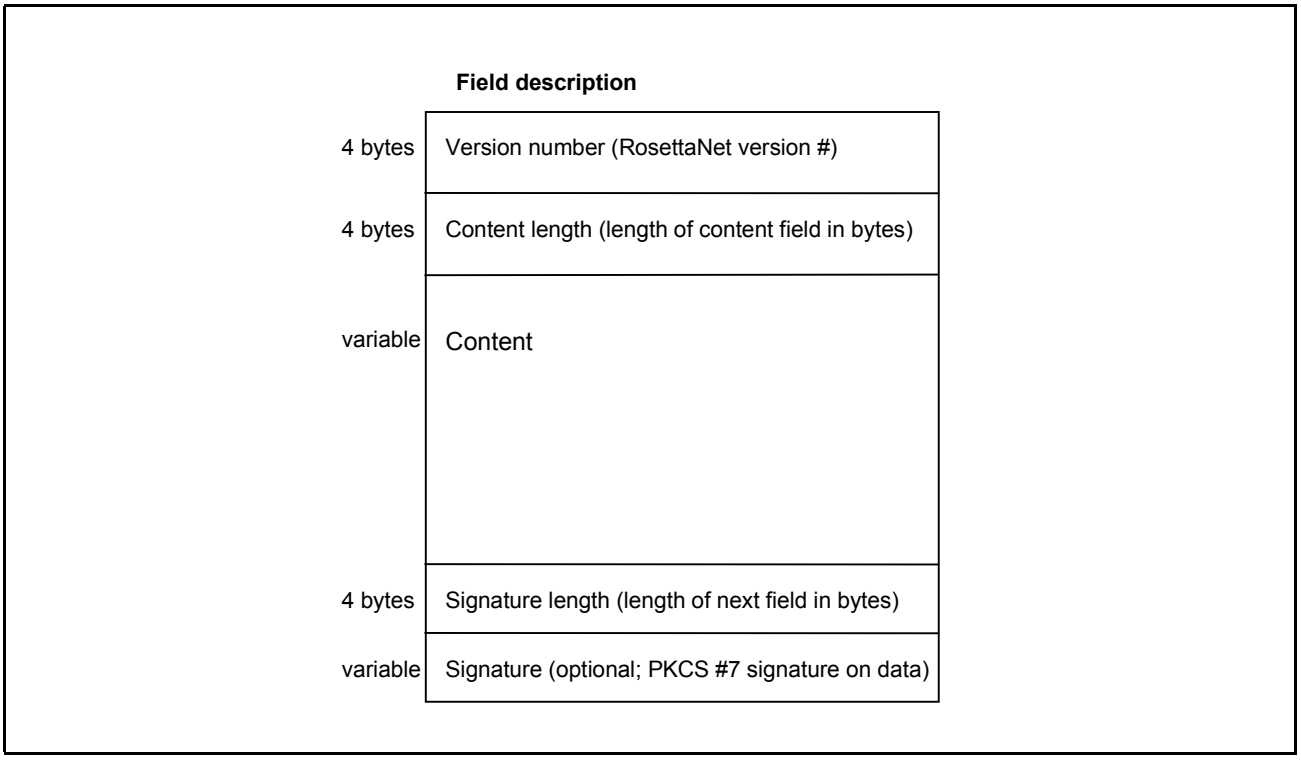


Figure 12. Structure of a RosettaNet Object

3.3.3 HTML User Agent Protocol

RosettaNet PIPs currently make use of Hypertext Markup Language (HTML) v3.2 as specified by the W3C.

The server-browser-server method for transmitting RosettaNet messages relies on the use of hidden fields within HTML forms and uses a user's browser during the transmission. Hidden values in an HTML form allow a CGI application to pass name-value pairs to a browser without the knowledge of the end user. Using the POST method, a browser can send name-value pairs from an HTML form to a CGI application via a Web server. The server then starts the designated CGI application and passes the browser-supplied data to the application.

RosettaNet specifies a parameter, ERNO, for passing the complete encoded RosettaNet Object to an HTML user agent. The HTML user agent protocol message encapsulates an upper

protocol message as a single name-value pair. This name-value parameter is described as follows:

Parameter: ERNO

Occurrences: Multiple

Usage: RosettaNet service to HTML user agent

Value: A base64 encoded RosettaNet Object containing encapsulated upper protocol message and an optional digital signature.

Example 5 shows how the name-value pair parameter is used in an HTML document instance. This message is returned as a response to a previous request for this message.

Example 5. Name-Value Pair Parameter

```
<HTML>
   <HEAD>
      <TITLE>Subscription Request</TITLE>
   </HEAD>
   <BODY>
      <FORM METHOD=POST ACTION="http://www.example.com/rn-agent/submit" >
      <INPUT TYPE=HIDDEN NAME=ERNO
VALUE="the_base64_encoded_RosettaNet_Object">
      <INPUT TYPE=SUBMIT NAME="Submit">
      </FORM>
   </BODY>
</HTML>
```

In this example, a "SUBMIT" button is placed in an HTML user agent window. When the button is pressed, a "POST" method on the network resource identified by the action URL is invoked. The name-value pair is then passed as a parameter of this method whose format is specified as a CGI protocol message (see following section for details). The RosettaNet Object is converted to a base64-encoded object before it is encapsulated as the value for the "VALUE" attribute.

3.3.3.1 Encoding RosettaNet Objects

When a RosettaNet object is transferred embedded in an HTML form, it must be base64 encoded as specified in RFC 1521. The base64 encoding method converts an arbitrary sequence of 8-bit bytes to a form that can be represented in 6-bit groups. In the encoding process, a sequence of three 8-bit bytes is treated as four 6-bit groups, each of which is then translated into a single digit in the base64 alphabet that uses a 65-character subset of ASCII. (This subset has the property that it is highly portable in that it is represented identically in all versions of ISO 646 including US-ASCII and in all versions of EBCDIC.) The output stream (encoded bytes) must be represented in lines of no more than 76 characters each. Base64-encoded data are about 33% larger than non-encoded data.

3.3.3.2 Compliance

1. The “POST” method must be specified in the message.
2. The value “ERNO” in the "NAME" field must be capitalized.
3. The RosettaNet Object must be base64 encoded prior to insertion into the value field for the “VALUE” attribute, when embedded in an HTML form.
4. Partners must agree on the URL value for the “ACTION” attribute.

3.3.4 Common Gateway Interface (CGI) Agent Protocol

The CGI agent protocol message is typically transmitted from a web browser to a RosettaNet service. It is not recommended that this protocol be used between services. The RosettaNet agent protocol should be used instead. However, CGI can be supported for limited or special use RosettaNet applications.

The CGI agent protocol comprises a name-value pair parameter with the same semantics of the HTML user agent protocol but with a different syntax. The name of the parameter for this protocol message is “ERNO” and the value of the parameter is a RosettaNet Object. The syntax for the name-value pair is as follows:

`ERNO=`*`RosettaNet_Object`*

3.3.4.1 Compliance

1. The keyword “ERNO” must be capitalized.

3.3.5 HyperText Transfer Protocols

The HyperText Transport Protocol (HTTP) is the core transport protocol used for the World Wide Web. RosettaNet implementations can use HTTP operating over TCP/IP for unsecured transfer of upper protocol messages or HTTP operating over SSL (HTTPS) for secured transfer of upper protocol messages.

3.3.5.1 Structure of an HTTP Message

This section, describing some of the structure of an HTTP request and response message, is adapted from the HTTP/1.0 specification. This section comprises only those parts of the HTTP specification that are used to specify the parameters required to transfer upper protocol messages passed down the RosettaNet protocol stack. Please refer to the HTTP specification for more details.

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta-information, and possible entity-body content.

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use. The following grammar rule specifies the format of this message.

```
Request       = Request-Line
                *( ...
                | entity-header )
                CRLF
                message-body
```

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. Space (SP) characters separate the elements. No CR or LF is allowed except in the final CRLF sequence. The following grammar rule specifies the format of the request line.

```
Request-Line   = Method SP Request-URI SP HTTP-Version CRLF
```

After receiving and interpreting a request message, a server responds with an HTTP response message. The following grammar rule specifies the format of the response message.

```
Response      = Status-Line
                *( ...
                | entity-header )
                CRLF
                message-body
```

The Status-Line consists of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence. The following grammar rule specifies the format of the status line.

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

The entity-header fields define meta-information about the entity-body or, if no body is present, about the resource identified by the request. Only those parameters used to transfer upper protocol messages from the RosettaNet protocol stack are shown. Please refer to the HTTP specification for more details.

```
entity-header  = Content-transfer-encoding
               | Content-type
               | Content-length
               | ...
```

Content-Type specifies the media type of the underlying data. Content-Encoding may be used to indicate any additional content encoding applied to the data in the message-body. The Entity-Length of a message is the length of the message-body before any transfer-encoding has been applied.

3.3.5.2 RosettaNet HTTP Format Specification

There are two RosettaNet HTTP "Request" message format specifications.

1. An HTTP Request message that is used to exchange a RosettaNet agent protocol message.
2. An HTTP Request message that is used to exchange a CGI agent protocol message.

The request line in the message must use the "POST" verb as the "Method" parameter for all request exchanges. Two MIME types are used to specify the content type of the message body.

- The MIME type **application/x-rosettanet-agent.**
- The MIME type **application/x-www-form-urlencoded.**

Example 6 shows the transfer-protocol request message for exchanging an encapsulated RosettaNet agent protocol message.

Example 6. RosettaNet Message Exchange

```
POST https://www.example.com/rn-agent/submit HTTP/1.0
Content-length:  3492
Content-type:  application/x-rosettanet-agent; version=1.0

RosettaNet_Object
```

Example 7 shows the transfer-protocol request message for exchanging an encapsulated CGI agent protocol message.

Example 7. CGI Message Exchange

```
POST https://www.example.com/rn-agent/submit HTTP/1.0
Content-length:  3492
Content-type:  application/x-www-form-urlencoded

ERNO=RosettaNet_Object
```

There is one RosettaNet "Response" message format specification that is used to exchange an HTML user agent protocol message. One MIME type is used to specify the content type of the message body.

- The MIME type **text/html.**

Example 8 shows the transfer-protocol response message for exchanging an encapsulated HTML user agent protocol message.

Example 8. HTML Message Exchange

```
HTTP/1.0 OK 200
Content-length:  3492
Content-type:  text/html

<HTML>
   <HEAD>
      <TITLE>Subscription Request</TITLE>
   </HEAD>
   <BODY>
      <FORM METHOD=POST ACTION="http://www.example.com/rn-agent/submit" >
      <INPUT TYPE=HIDDEN NAME=ERNO
          VALUE="the_base64_encoded_RosettaNet_Object">
      <INPUT TYPE=SUBMIT NAME="Submit">
      </FORM>
   </BODY>
</HTML>
```

3.3.5.3 Compliance

- Implementations must support HTTP/v.1.0 or higher
- If using MIME types, then one of the following must pertain, as appropriate:
  - Use of the MIME type **application/x-rosettanet-agent.**
  - Use of the MIME type **application/x-www-form-urlencoded.**
  - Use of the MIME type **text/html.**
- The "POST" method for HTTP requests.
- Content encoding value must be "base64" when transferring RosettaNet agent protocol messages embedded in an HTML form.

3.3.6 Secure Socket Layer (SSL) Protocols

Refer back to Table 1 for terminology. Transmission of RosettaNet Objects using HTTP makes use of the Secure Sockets Layer (SSL) Protocol that is being standardized by the Internet Engineering Task Force (IETF). This security protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. RosettaNet specifies use of SSL Version 3 and requires use of the "client authentication" mode of SSL where the client and server authenticate each other. Authentication between parties is based on the exchange of valid certificates.

Use of the SSL v.3 protocol in the transmission of RosettaNet Objects provides data integrity, mutual authentication, and confidentiality. Specifically, it ensures that:

- the object sent was properly received and not tampered with during transmission
- the sender has cryptographic assurance of the identity of the recipient
- the recipient has cryptographic assurance of the identity of the sender
- transmission is encrypted so that third parties eavesdropping on network components may not view the contents of RosettaNet Objects being exchanged

SSL V3 includes the real-time negotiation of algorithms for encryption and message integrity between client and server. Standard SSL ciphers include DES, 3DES, RC4, and RC2. Key lengths used for encryption during SSL sessions are related to the selected cipher suite. If the negotiated cipher is based on RC4, the key length will be either 40 or 128 bits. For DES the key length will be 56 bits and for 3DES the key length will be 168 bits.

Partners are responsible for negotiating SSL ciphers and key lengths for RosettaNet transactions. In all cases however the selected cipher suite must provide at least 40-bit encryption and must ensure that authentication occurs, i.e. Diffie-Hellmann without certificates should not be used.

4 Digital Signatures

The digital signature specification is based on the use of PKCS #7 detached signatures which are described in *PKCS #7: Cryptographic Message Syntax Standard* by RSA Data Security. PKCS #7 describes a general syntax for data with cryptography applied to it. This specification is compatible with PKCS #7 in that it is based on the data type for signed data defined by PKCS #7. This section specifies the use of PKCS #7 signatures within RosettaNet-compliant systems to ensure interoperability across implementations.

Digital signatures are specified as "detached signatures" which means that the digital signature is encoded separately from the data that has been signed. The primary benefit of using a detached signature is that it enables implementations that do not support digital signatures to handle the upper protocol message that is signed.

The transport security protocol used to transmit RosettaNet Objects also provides the recipient with a high level of assurance of origin and content. The Secure Sockets Layer authenticates the sender and receiver and protects the integrity and confidentiality of the data stream. The additional benefits of a digital signature must be compared with the cost of implementation. RosettaNet implementations and trading partner relationships may decide to forego the use of digital signatures.

Descriptions of the data structures and the processes involved in constructing and verifying a PKCS #7 detached signature follow to illustrate the use of PKCS #7 within RosettaNet-compliant applications. Implementers should refer to the PKCS #7 document for correct grammar and syntax.

Use of digital signatures is optional by trading partners engaged with specific PIPs. Note that some PIPs require the use of digital signatures. Also, solution partners wishing to support multiple PIPs among arbitrary trading partners should make provision for supporting the use of digital signatures. When used, digital signatures are based on the PKCS #7 cryptography standard and are detached signatures encoded separately from the data to which they apply. If a signature is used within a RosettaNet Object it is a signature on the data contained within the RosettaNet Object's content field. The primary benefit of a detached signature is that implementations that do not support digital signatures will be able to process data that has been signed even though they are unable to verify the signature.

Within RosettaNet, signatures are associated with Initiating and Servicing Organizations and not with individuals. A digital signature provides non-repudiation of origin and content for the transmission of the object between partners in the supply chain. The recipient is assured that a particular message was sent by a service or someone with access to the signer's private key. This means a message cannot be forged or tampered with and a signer cannot deny having sent a particular message.

It should be noted that the transport security protocol used to transmit RosettaNet Objects also provides the recipient with a high level of assurance of origin and content as used here. The Secure Sockets Layer authenticates the sender and protects the integrity of the data stream. If the result of the authentication step is passed along to the application, then this provides the recipient with information that can be used to ensure the authenticity of the received order.

Digital signatures can provide two additional advantages. First, they provide evidence that can be stored for later retrieval if non-repudiation is a partner interface process requirement. Second, they provide the opportunity to have multiple authorized signers within an organization. However, these additional benefits of digital signatures must be compared with the cost of implementation. Initial RosettaNet implementations may choose to forego the use of digital signatures.

4.1 Structure of a RosettaNet Digital Signature

The structure of a RosettaNet Object's signature, based on PKCS#7, is a BER-encoded object that consists of the fields and values in Table 3.

Table 3. Fields within a RosettaNet Digital Signature

Name	Value	Description
ContentType	SignedData	Content Type
Version	1	Current syntax standard
DigestAlgorithms	md5 and/or sha-1	Identifies the algorithms used in the signatures
ContentInfo		This field is empty when a detached signature is used
Certificates		Field for transmitting signer's certificate(s)
IssuerAndSerialNumber		The unique identifier of the signer's certificate and public key
DigestAlgorithm	md5 or sha-1	Identifies algorithm used
DigestEncryptionAlgorithm	RsaEencryption	Digest Encryption Algorithm
EncryptedDigest		This is the digital signature on the external upper protocol message.

4.2 Preparing and Verifying a Signature

The process of preparing a PKCS #7 digital signature breaks down into the following steps:

1. Encoding a value of type ContentInfo according to PKCS #7 for the data to be signed.

   This step is not necessary when a detached signature is used (as specified in RosettaNet) as the inner ContentInfo field is left empty. It is included here for completeness.

2. Digesting the data to be signed according to PKCS #7.

   This will be done with the signer's message digest algorithm; either MD5 or SHA-1 is required. The input to the message digest algorithm is the original data content. The output, the message digest, is an octet string.

3. Encoding a value of type DigestInfo according to PKCS #7 from the message digest. Digesting the data to be signed according to PKCS #7.

The output of this step is a BER-encoded value which includes the digest algorithm identifier and the message digest.

4. Encrypting the encoded DigestInfo value with the signer's private key.

   This will be done with RSA. The result is an octet string representing the encryption with the signer's private key of the BER encoding of a value of type DigestInfo. This is often referred to as the encrypted message digest.

5. Encoding a value of type SignedData according to PKCS #7 from the first ContentInfo value, the encrypted message digest, and other information.

   The first ContentInfo value will be empty since RosettaNet specifies the use of detached signatures. The output of this step is a BER-encoded value which includes the encrypted message digest, the signer's certificate(s), the unique identifier of the signer's certificate, the message-digest algorithm identifier, and the message digest encryption algorithm.

6. Encoding a value of type ContentInfo according to PKCS #7 from the SignedData value.

   The content type is PKCS #7 SignedData. The output of this step is a BER-encoded value which includes the SignedData value.

7. The original order content and the BER-encoded PKCS #7 value of type ContentInfo are placed in an RosettaNet Object.

8. The order content is placed in the RosettaNet content field. The ContentInfo value is placed in the signature field.

The process of verifying a signature in a RosettaNet Object breaks down into the following steps:

1. Extracting the BER-encoded PKCS #7 value of type ContentInfo from the signature field of the RosettaNet Object.

2. Decoding the value of type ContentInfo to extract the SignedData value.

   The content type should be verified to be PKCS #7 SignedData. The output of this step is the BER-encoded value SignedData.

3. Decoding the SignedData value to determine the signer's public key, the encrypted message digest, and the message digest encryption algorithm.

   The output of this step is an identifier that uniquely identifies the sender's public key, an identifier that identifies the digest encryption algorithm and the encrypted message digest. The signer's public key is contained in a certificate included in the SignedData and is referenced by an issuer name and serial number that uniquely identifies the certificate for the public key.

4. Decrypting the encrypted message digest with the signer's public key and the message digest encryption algorithm identified in the SignedData value (RSA for RosettaNet) to obtain a value of type DigestInfo.

   The output is a BER-encoded value which includes the digest algorithm identifier and the recovered message digest.

5. Decoding the value of type DigestInfo to obtain the message digest and the digest algorithm identifier.
6. Digesting the contents of the RosettaNet data field with the digest algorithm identified in DigestInfo.

   The output of this step is the computed message digest which is compared to the recovered message digest from the prior step to verify the signature.
7. Verifying the signature by comparing the recovered message digest to the computed message digest.
8. Verifying signer's certificate.
9. Verifying that signer is an authorized signer for this trading partner.

4.3 Compliance

Compliance with RosettaNet digital signature specifications requires the following:

1. If a signature is used, the signature field within a RosettaNet Object must contain a PKCS #7 *ContentInfo* data object of type *SignedData*, encoded using the Basic Encoding Rules;
2. If a signature is used, it must be a PKCS #7 detached signature which means that *SignedData* will contain a signature on the external upper protocol content and its inner *ContentInfo* field will be empty;
3. Support for MD5 and SHA-1 message digest algorithms;
4. Support for RSA encryption for digest encryption;
5. Support for verification of signatures using RSA public key sizes from 512 (minimum) to 1024 (recommended) bits.

5 Authentication Using SSL and Digital Certificates

The RosettaNet authentication model is based on the use of the Secure Sockets Layer (SSL) v.3 protocol and RosettaNet digital certificates to ensure that:

- the sender has cryptographic assurance of the identity of the recipient;
- the recipient has cryptographic assurance of the identity of the sender.

Authentication is the process of reliably establishing the identity of a party that is communicating.

At a high-level, SSL authentication works as follows. SSL-enabled servers typically accept SSL connection requests from clients on port 443. When the client connects to this port, it initiates an SSL "handshake" which establishes the session. During this handshake, the authentication process begins with an exchange and verification of certificates between the client and server. When mutual authentication is used (as specified in RosettaNet), each party must present a

verifiable certificate from a trusted certificate authority. If either party cannot, the handshake fails and the session terminates.

There are several aspects to certificate verification (and authentication) within SSL. First, each party must prove it is the legitimate owner of the certificate it presents. The certificate itself does not authenticate, the combination of the certificate and the correct private key does. To demonstrate that the entity presenting the certificate is the legitimate certificate owner, SSL requires that the presenter digitally sign data exchanged during the handshake. Each party signs protocol data (including its certificate) to prove it is the legitimate owner of the certificate. Certificates are also verified by checking their validity dates and by verifying the digital signature of the trusted certificate authority which is included with the certificate.

Once authentication has occurred, a server can map the client's name in the certificate to access control databases. In this way, end-users present their certificates rather than usernames and passwords to gain access to information that is access-controlled. Within RosettaNet, this is the model assumed for user access to private services containing confidential information. In addition, information from the verified RosettaNet certificate is used to identify the user within messages transmitted from Servicing Organizations to Initiating Organizations.

5.1 Digital Certificate Specifications

A digital certificate is an electronically signed document issued by a trusted third party, called a Certificate Authority, which binds identifying information to an individual's public key. Within RosettaNet, X.509 certificates are used by SSL for authentication of both users and servers. Specifically, certificates are used by:

- a user, to authenticate to Servicing Organization service sites
- Initiating Organization server to authenticate to Servicing Organization servers
- Servicing Organization server to authenticate to users and to Initiating Organization servers

The RosettaNet specification defines minimal requirements for public key certificates for users and servers with the goal of simplifying implementation and achieving interoperability. These requirements are outlined in the following sections.

Digital certificates may also be optionally used within RosettaNet to securely distribute the public keys that trading partners use to verify a company's digital signature on a signed order or order request document. The certificate used to distribute the public key associated with a digital signature on an message will typically be a different certificate (with a different public key) than the one used by the same trading partner for authentication during SSL sessions. It is distributed with the digital signature.

5.1.1 Certificate Content Requirements

Digital certificates contain data that identify the holder of the certificate. Certificates for use in RosettaNet applications are based on the X.509, Version 3 standard. The X.509 standard defines the data that can be used within a certificate. RosettaNet specifies a subset of these fields that are the minimum required content for certificates used within RosettaNet-compliant

systems. Other fields may be included in the certificate but the following fields (with exception of Email address) are required for RosettaNet applications:

- Certificate Serial Number
- Subject Common Name
- Subject Organization Name
- Subject Email address (optional)
- Subject Public Key
- Certificate Validity Period
- Issuer Organization Name
- Issuer Signature

RosettaNet recommends that user certificates contain data in the Subject field that uniquely identifies the user. For the purposes of RosettaNet, the Subject Common Name and/or the Subject Email address (if available) in combination with the Organization Name are the certificate fields used to establish the identity of the user.

RosettaNet recommends that the user certificate contain data that uniquely identifies the organization with which the user is affiliated in order that a Servicing Organization provide the user with access to the appropriate service information. The Subject Organization Name in the user certificate establishes the identity of the initiating organization. An Initiating Organization is responsible for keeping the use of its "organization name" consistent across its user certificates so that trading partners can use the contents of this field to authorize service access.

If the Organization Name field is not sufficiently granular to establish the appropriate service functionality, trading partners can agree to use additional mechanisms (in conjunction with certificates for authentication) to convey necessary information (e.g. an account number) as part of the user's access to the service.

5.1.2 Selection of a Certificate Authority

The value of a certificate lies in the fact that the integrity of the data contained in the certificate can be cryptographically verified. This process involves verifying the validity of a chain of certificates up to a trust point or Root Certificate. Each certificate in the chain is linked to the one above through the use of digital signatures. Verification involves the validation of these links starting at the bottom. Validation stops when the trust point or Root Certificate is reached. The Root Certificate is a self-signed certificate generated by a Certificate Authority (CA). It is essential that the Root Certificate be distributed in a secure manner, from a secure source and stored locally for use in validating certificates.

Initiating Organizations implementing RosettaNet-compliant solutions will need to decide whether to outsource digital certificate services to an external, 3rd party Certificate Authority (CA) or to use an internal certificate infrastructure based on a company-specific CA. Many factors must be considered in making this decision including the corporation's internal

information technology capabilities and strategy, cost, time-to-market, interoperability considerations, trading partner plans, etc.

An important factor in the selection of a CA for use in RosettaNet (or other inter-enterprise) applications is that the CA's Root Certificate be distributed widely, in a secure manner. Optimally, the CA's Root Certificate will be distributed with standard Web browser software. This will tend to minimize the implementation issues associated with certificates.

RosettaNet recommends the use of a commercial 3rd party CA whose Root Certificate is included within common browser software as long as the CA can meet the RosettaNet content requirements for certificates as specified in the previous section.

Initiating organizations may, in the future, when an appropriate infrastructure is in place, use self-issued certificates (certificates generated and signed by the organization itself rather than a trusted 3rd party) in RosettaNet applications as long as these certificates meet the minimal certificate content requirements specified in the previous section. However, RosettaNet discourages the use of self-issued certificates because these will increase the burden of trading partner set-up and maintenance. Initiating organizations that leverage internal self-issue certificate infrastructures in RosettaNet applications must plan to issue (and re-issue as needed) certificates to trading partners and to provide trading partners with a copy of their Root Certificate.

For administrative convenience of other trading partners, servicing organizations MUST install site certificates issued by a commercial 3rd party CA whose Root Certificate is distributed with common browser software. This is important to avoid the need to update user browsers with new Root Certificates when setting up a new trading partner. If an Initiating Organization uses self-issue certificates, the Servicing Organization will also need to install the Initiating Organization's Root Certificate.

5.1.3 Certification Classes and Policies

Commercial Certificate Authorities offer varying levels of service based on customer requirements and intended use of the certificates. These levels of service affect certificate issuance, management and revocation as well as operational controls and assurances that are provided by the CA. For example, the process a particular CA uses to verify the information contained in a digital certificate prior to issuance, such as information related to the individual's identity, may vary depending on the level of service. Applications involving large financial transactions may require extensive verification of information while applications with limited security needs may require only limited verification.

5.1.4 Certificate Revocation Lists

There will occasionally be a need to revoke a digital certificate. This might happen, for example, as the result of a user leaving the company or changing job responsibilities. Revocation of a certificate is the responsibility of the organization and its issuing CA. Some CAs publish a list of revoked certificates known as a certificate revocation list (CRL) on a regular basis that can be used for access control purposes.

Organizations are responsible for informing their trading partners of revocations of server certificates in a timely manner.

5.1.5 Compliance

Compliance with the RosettaNet authentication model requires that:

- browsers and servers must support SSL V3
- browsers and servers must be able to present verifiable certificates to authenticate themselves
- certificates must meet RosettaNet certificate content requirements
- browsers and servers must be able to use certificates to authenticate the other party
- SSL mutual authentication is required
- the negotiated cipher suite must not allow a non-authenticated session (i.e. no Diffie-Hellman without certificates)
- RSA public key length be 512 to 1024 bits.
- the authentication step must provide applications with certificate information necessary for authorization decisions including the Subject Common Name, Subject Organization Name, Issuer Organization Name, etc.

6 Technical Compliance

In order to promote interoperability it is necessary to define the concept of RosettaNet technical compliance. RosettaNet technical compliance specifies a *minimal* level of implementation that allows for the useful interoperability of systems and business processes. This section defines the requirements for such minimal technical compliance.

Technical compliance with this RosettaNet specification is outlined in two dimensions. First, all implementations must meet certain minimal requirements related to data, transport and security to be considered RosettaNet-compliant. Second, each of the interested parties (i.e. Servicing Organizations, Initiating Organizations and users) must meet certain minimal requirements to be considered technically compliant.

6.1 Compliance with PIP Specifications

Technical compliance with PIP specifications entails different compliance requirements for action, transaction, and process specifications.

Service Specification: Verify that content and sequence within the message is valid with respect to the Service Protocol Message DTDs and guidelines.

6.2 Compliance with Protocol Message Specifications

Implementations that are compliant with RosettaNet protocol message specifications MUST:

1. be able to create RosettaNet protocol messages formatted according to this RosettaNet specification;
2. be able to package upper protocol messages as RosettaNet Objects prior to transmission;
3. be able to interpret and process RosettaNet Objects without digital signatures;
4. be able to interpret and process RosettaNet Objects that contain digital signatures even if signature verification is not supported.

6.3 Compliance with Transfer-Related Specifications

Implementations compliant with RosettaNet specifications for server-to-server transfer MUST:

1. be able to transmit RosettaNet Objects directly to RosettaNet-compliant servers using HTTP POST with SSL v.3 as specified in this RosettaNet specification;
2. be able to receive RosettaNet Objects directly from RosettaNet-compliant servers using HTTP POST with SSL v.3 as specified in this RosettaNet specifications;
3. be able to recognize and interpret the Content-Type "application/x-rosettanet-agent";
4. be able to designate a URL path where RosettaNet Objects can be received;
5. be able to provide information from authentication session to applications for verification of message origin.

Implementations compliant with RosettaNet transfer-related specifications for server-browser-server transfer MUST:

1. if sending, be able to use HTTP POST method to transmit base64-encoded RosettaNet Objects to a known URL path via a browser using a hidden field ("ERNO") in an HTML form as specified in this RosettaNet specification;
2. if receiving, be able to receive base64-encoded RosettaNet Objects at a designated URL path via an HTTP POST from a browser using a hidden field ("ERNO") in an HTML form, as specified in this RosettaNet specification.

6.4 Compliance with Security-Related Specifications

Implementations compliant with RosettaNet security-related specifications MUST:

1. be able to use SSL v.3 protocol for secure Internet communications;
2. be able to use the mutual authentication mode of SSL for authentication between clients and servers;

3. be able to use (at minimum) 40-bit encryption for SSL sessions;
4. be able to use certificates for authentication of clients and servers as specified by this RosettaNet specification;
5. be able to use authentication information for access control;
6. NOT provide, or require the use of, digital signatures that are not in compliance with RosettaNet technical specifications.

Note that minimal compliance with RosettaNet security-related specifications DOES NOT REQUIRE:

1. that clients and servers support certificate revocation lists as part of authentication;
2. that clients and servers be able to include RosettaNet-compliant digital signatures within a RosettaNet Object;
3. that clients and servers be able to verify RosettaNet-compliant digital signatures contained within a RosettaNet Object.

6.5 Technical Compliance for Servicing Organizations

A Servicing Organization that is compliant with RosettaNet technical specifications MUST:

1. be able to authenticate uses prior to service access through the use of digital certificates consistent with this RosettaNet specification;
2. be able to limit user access to private services based on information contained in digital certificates and optionally through profile information presented at time of service access;
3. be able to create RosettaNet Objects containing upper protocol messages consistent with this RosettaNet specifications;
4. be able to send RosettaNet Objects containing upper protocol messages via the Internet to RosettaNet-compliant trading partner servers using server-to-server transport or server-browser-server transport;
5. designate a URL at which it can receive RosettaNet Objects containing messages from trading partners;
6. be able to receive RosettaNet Objects containing upper level protocol messages via the Internet from RosettaNet-compliant trading partners servers consistent with this RosettaNet specification;
7. be able to present a valid certificate consistent with RosettaNet specifications for use in authentication during interactions with trading partners;
8. be able to authenticate trading partner servers that present valid digital certificates;
9. be able to support secure Internet communications through SSL v.3 Internet security protocol.

6.6 Technical Compliance for Initiating Organizations

An Initiating Organization that is compliant with this RosettaNet specification MUST:

1. provide users with Internet access to services located at the Servicing Organization site;
2. enable secure Internet communications by supporting use of SSL v.3 Internet security protocol by users and servers across corporate firewalls;
3. publish a URL at which it can receive service requests from trading partners;
4. be able to receive messages at this URL via the Internet from RosettaNet-compliant trading partner servers through either the server-to-server transport method or the server-browser-server method;
5. be able to create RosettaNet Objects containing messages consistent with RosettaNet specifications
6. be able to send RosettaNet Objects containing messages to trading partner servers via the Internet consistent with RosettaNet specifications;
7. be able to present a valid certificate compliant with this RosettaNet specification for use in authentication during interactions with trading partner servers;
8. be able to authenticate trading partner servers that present valid digital certificates.

6.7 Technical Compliance for Users

A user that is compliant with RosettaNet technical specifications MUST:

1. have a workstation with Internet connectivity;
2. have a secure Web browser (such as Netscape Navigator 3.0 or later or Microsoft Internet Explorer 3.0 or later) installed on workstation;
3. have a valid certificate compliant with this RosettaNet specification securely installed in the browser for use in authentication with servicing organization service sites;
4. be able to use SSLv.3 for secure Internet communications.

6.8 Technical Compliance for Third Party Agents

A third party agent that is compliant with RosettaNet specifications MUST:

1. comply with RosettaNet specifications (as stated above) for the entity for which the agent is acting as proxy.

Note that third party agent solutions that meet the above condition are said to be technically compliant and are assumed to be capable of inter-operating with other RosettaNet-compliant solutions.

6.9 Technical Compliance For Technology Provider Solutions

A technology provider solution that is compliant with this RosettaNet specification MUST:

1. be able to create messages consistent with RosettaNet specifications;
2. be able to send messages over the Internet consistent with this specification;
3. be able to receive messages via the Internet from other RosettaNet-compliant solutions consistent with this specification;
4. be able to interpret messages correctly;
5. enable the location of services at Servicing Organization's site;
6. enable the location of user profile information at Initiating Organization's site;
7. be able to accept certificates for authentication and access control for servers and users consistent with this specification;
8. be able to limit access to sensitive information based on the authenticated identity;
9. support SSL v.3 for secure Internet communications;
10. comply with this RosettaNet specification.

Note that vendor solutions that meet the above conditions are said to be technically compliant and are assumed to be capable of inter-operating with other RosettaNet-compliant solutions. RosettaNet has no requirement for third party validation. Some parties may wish to offer or accept such a service.

7 Implementation Needs

This section provides details for implementing some of the technical requirements specified in earlier sections.

7.1 RosettaNet Protocol Stack Specification

The following notes relate to the transmission of RosettaNet Objects using Server-to-Server HTTP Protocol:

1. The decision regarding which approach to implement for transmission of RosettaNet Objects will be based on several factors including the desired user's experience, the internal processes of the partner organizations, security requirements, error handling, etc. Which method is used between two partners should be discussed and agreed upon between partners. To insure interoperability across a variety of partner systems, partner implementations will need to be capable of supporting both server-to-server and server-browser-server methods for upper protocol message transport.
2. Port 443 is reserved for use of HTTP with SSL.

3. The application sending messages should maintain a queue of pending RosettaNet Objects to send. The sender should remove objects from the queue only after the recipient has acknowledged successful receipt with a 200 response.
4. The authenticated name of the transaction initiator should be passed from the transport layer to the recipient's application so that the application will be able to verify (in the absence of a digital signature) that the initiator was authorized to send the actual message that was received. Otherwise, it would be possible for an entity with a legitimate RosettaNet certificate to send an unauthorized message under another entity's name.
5. It is not the responsibility of the transport layer to detect duplicates.

The following notes relate to the Server-Browser-Server Method for transmitting RosettaNet Objects:

1. This method of transmission is less robust than the server-to-server method specified in section 3.2.1 of this specification but it does have the advantage of keeping the user "in the loop" as a message is transmitted from one service to another. The browser/user will be a potential intermediate point of failure. If the workstation crashes before the transmission is complete, the transaction will be left in an undefined state. The POST may have successfully completed or it may not. In this case, the user will need to determine whether the Servicing Organization server received the message and if not, must return to the Servicing Organization site to recreate the message. Since this is not a direct transmission from server to server, there is no opportunity to automatically retransmit from the Servicing Organization server if there is no acknowledgment from the Initiating Organization server. Also, if the user does recreate the message the Initiating Organization server will not be able to automatically screen for duplicates since the Servicing Organization will generate a new and unique transaction ID number for the recreated message.
2. Transmission of messages via hidden fields in HTML forms is less secure than the server-to-server method specified in section 3.2.1 of this specification. In particular, the Initiating Organization server will not be able to authenticate the Servicing Organization server since they are not establishing a direct connection. This means that an individual who knows how to construct and transmit RosettaNet Objects to the fictitious company, MegaCorp, could submit a forged message via the browser of an unsuspecting MegaCorp user who happened to be browsing the network at the wrong place and time. This might not be caught except by the user. Although it is highly unlikely that one of MegaCorp's trading partners would fake message, a malicious third party who wanted to create confusion could forge a message exchange under the name of one of MegaCorp's trading partners. If this is a concern, MegaCorp could require its trading partners to attach digital signatures to RosettaNet Objects.

The following notes relate to the use of the Secure Sockets Layer protocols:

1. It is possible to maintain an SSL session in order to avoid repeating the complete SSL authentication each time. This is recommended where appropriate.
2. Encryption algorithms cannot currently be exported if key lengths are greater than 40 bits although it may be possible to obtain a waiver based on "use for financial purposes".

Default browser and server installations typically support 40-bit SSL encryption that has been shown to be breakable by code crackers who have sufficient time and access to computing resources. Typically, to obtain more secure 56- or 128-bit SSL encryption requires updating both browsers and servers.

7.2 Digital Signatures

1. Except for ContentType and Content, the actual object identifiers or values for the fields are not specified in PKCS #7. See PKCS #1, PKCS #9, and *S/MIME Implementation Guide* (Version 2) for these object identifiers.
2. PKCS #7 places no requirements on the format of the data that is signed. The RosettaNet data to be signed is the agent transaction protocol.
3. There is a degenerate case of SignedData in which there is no signature included. This can be used for disseminating certificates and certificate revocation lists. This is not used within RosettaNet.
4. PKCS #7 allows for countersignatures, i.e. the data to which the signature applies is itself signed data. This is not supported within RosettaNet since the detached signature will apply to the content of the RosettaNet data field in the RosettaNet Object and this is defined to be a string representing the data in the *content* field.
5. BER encoding is defined in I.T.U. X.209. The BER were developed as a companion standard to ASN.1. These rules take an ASN.1 description and derive a transfer representation based on a tag, length, value scheme. The BER allow the automatic derivation of a transfer syntax (e.g. hexadecimal 21) for every abstract syntax defined using ASN.1.
6. The data produced by BER encoding is 8-bit binary data. The entire RosettaNet Object including the BER-encoded signature is converted to base64 prior to HTTP transport to ensure that the RosettaNet Object is transferred intact.
7. The following optional PKCS #7 SignedData fields are not specified for use within RosettaNet implementations: crls, authenticated attributes, and unauthenticated attributes. Attributes would allow other information such as time stamps to be included in the signature but use of attributes is not specified for RosettaNet.
8. Developer toolkits, such as Microsoft's CryptoAPI and RSA Data Security's BSAFE, provide services that enable application developers to add PKCS cryptography including digital signatures to their applications.
9. To be able to prove non-repudiation to a third party at a later date, implementers should securely store the following items for later retrieval: the RosettaNet Object including the digital signature, the signer's certificate and the appropriate Certificate Authority's public key, the current certificate revocation list, and proof of date and time. Note that this applies to logging requirement in server-to-server message transfer. There is no logging at the browser.

7.3 Authentication Using SSL and Digital Certificates

1. The RosettaNet specification does not address how a user authenticates to servers within the Initiating Organization because this does not affect interoperability.
2. Partners must configure their servers such that certificate information obtained by SSL during authentication is presented to applications as CGI environment variables or via server APIs.
3. The messages sent from Servicing Organizations to Initiating Organizations must include user identity information obtained during the authentication step that the Initiating Organization can map to its user profile database. At a minimum this information will include Subject Common Name from the certificate. However, this may not uniquely identify the user, so trading partners can optionally agree to have Initiating Organization transmit a unique user ID prior to service access. When this ID is provided the Servicing Organization will include this in the exchanged message. Otherwise, the Servicing Organization will include the Subject Electronic Mail address from the certificate if that is present.
4. A user's digital certificate is typically stored within the browser on the user's desktop computer. A user will need to obtain a second certificate for an additional computer at home or in the office. It should be possible to obtain multiple certificates, i.e. certification for the same name with multiple key pairs, from the Certificate Authority. In this situation, the certificate's serial number will distinguish each certificate. Nominally certificates are portable (e.g. the use of crytographically enabled "Smart Cards") but current implementations and operational restrictions limit this portability.
5. If the user has multiple certificates installed, it will be necessary to configure the browser to specify which certificate will be used.
6. A user's private key MUST be password protected.
7. Initiating and servicing organizations will maintain the certificate Issuer Organization Name and the certificate Subject Organization Name in a trading partner database to assist in the authentication and authorization process.
8. The servicing organization must be able to associate the Subject Organization Name used with user certificates with the Initiating Organization's entry in its trading partner database. This organization name should be exchanged as part of trading partner set-up. The initiating organization is responsible for insuring the consistent use of this name across all its user certificates. This will enable Servicing Organizations to use information in a user's certificate to control access to private service functions. Use of a DUNS number in the organization name field of the user certificates is one mechanism for insuring uniform naming of the organization but is not required.

8 RosettaNet Protocol Message DTDs

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

guidelines and DTDs are listed as follows found via the web at http://www.commercedesk.com/rosettanetrepository.

- Preamble Part Message Guideline (PreamblePartMessageGuideline.html; PreamblePartMessageGuideline.dtd)
- Acceptance Acknowledgement Guideline (AcceptanceAcknowledgementGuideline.html; AcceptanceAcknowledgementGuideline.dtd)
- Acceptance Acknowledgement Exception Guideline (AcceptanceAcknowledgementExceptionGuideline.html; AcceptanceAcknowledgementExceptionGuideline.dtd)
- Exception Guideline (ExceptionGuideline.html; ExceptionGuideline.dtd)
- Receipt Acknowledgement Guideline (ReceiptAcknowledgementGuideline.html; ReceiptAcknowledgementGuideline.dtd)
- Receipt Acknowledgement Exception Guideline (ReceiptAcknowledgementExceptionGuideline.html; ReceiptAcknowledgementExceptionGuideline.dtd)
- Service Header Part Message Guideline (ServiceHeaderPartMessageGuideline.html; ServiceHeaderPartMessageGuideline.dtd)

Elements and attributes are defined in the RosettaNet Business Dictionary.

9 Complete Example of a Service Protocol Message

```
MIME-Version: 1.0
     Content-Type: Multipart/Related; boundary="RN-part-boundary";
                      type="Application/x-RosettaNet"
     Content-Description: This is the RosettaNet business message

     --RN-part-boundary
     Content-Type:  Application/XML; RNSubType="preamble-header"
     Content-Description: This is the Preamble Header part of the business
        message

<?xml version="1.0"  encoding="UTF-8"?>

<!DOCTYPE Preamble SYSTEM "PreamblePartMessageGuideline.dtd" >

<Preamble>

   <VersionIdentifier>1.01</VersionIdentifier>

   <DateTimeStamp>19990531T132000.0500Z</DateTimeStamp>

   <GlobalAdministeringAuthorityCode>RosettaNet
       </GlobalAdministeringAuthorityCode>

   <GlobalUsageCode>Test</GlobalUsageCode>

</Preamble>
```

```
     --RN-part-boundary
     Content-Type:  Application/XML; RNSubType="service-header"
     Content-Description: This is the Service Header part of the business
        message

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE ServiceHeader SYSTEM "ServiceHeaderPartMessageGuideline.dtd">

<ServiceHeader>

   <ProcessControl>

      <ProcessIdentity>

         <GlobalProcessCode>Distribute Product Stock Keeping Unit
             </GlobalProcessCode>

         <VersionIdentifier>1.0</VersionIdentifier>

         <GlobalProcessIndicatorCode>2A8</GlobalProcessIndicatorCode>

         <initiatingPartner>

            <GlobalBusinessIdentifier>123456789</GlobalBusinessIdentifier>

         </initiatingPartner>

<instanceIdentifier>

            <FreeFormText>1</FreeFormText>

</instanceIdentifier>

<description>

<FreeFormText xml:lang="en">The purpose of this PIP is to specify the process
for Buyers to notify customer managers when they have created a Stock Keeping
Unit (SKU) for their product.  This process is typically executed after the
Buyer has been informed about new products.</FreeFormText>

</description>

</ProcessIdentity>

      <ServiceRoute>

         <toService>

            <BusinessServiceDescription>

               <GlobalBusinessServiceCode>Buyer Service
                  </GlobalBusinessServiceCode>

            </BusinessServiceDescription>

         </toService>

         <fromService>

            <BusinessServiceDescription>

               <GlobalBusinessServiceCode>Customer Manager Service
                   </GlobalBusinessServiceCode>
```

```
      </BusinessServiceDescription>
   </fromService>
</ServiceRoute>
<TransactionControl>
   <GlobalTransactionCommandCode>Send</GlobalTransactionCommandCode>
   <AttemptCount>1</AttemptCount>
   <TransactionIdentity>
      <GlobalTransactionCode>Distribute Stock Keeping Unit
          </GlobalTransactionCode>
      <instanceIdentifier>
         <FreeFormText>1</FreeFormText>
      </instanceIdentifier>
      <description>
         <FreeFormText xml:lang="en">Activity distributes a Stock
             Keeping Unit (SKU) notification.</FreeFormText>
      </description>
   </TransactionIdentity>
   <PartnerRoleRoute>
      <fromRole>
         <PartnerRoleDescription>
            <GlobalPartnerRoleClassificationCode>Buyer
                </GlobalPartnerRoleClassificationCode>
         </PartnerRoleDescription>
      </fromRole>
      <toRole>
         <PartnerRoleDescription>
            <GlobalPartnerRoleClassificationCode>Customer Manager
               </GlobalPartnerRoleClassificationCode>
         </PartnerRoleDescription>
      </toRole>
   </PartnerRoleRoute>
   <ActionControl>
      <ActionIdentity>
         <GlobalBusinessActionCode>Stock Keeping Unit Creation
             Notification</GlobalBusinessActionCode>
         <VersionIdentifier>1.0</VersionIdentifier>
```

```
                <instanceIdentifier>

                   <FreeFormText>1</FreeFormText>

                </instanceIdentifier>

                <description>

                   <FreeFormText xml:lang="en">A notification informing the
customer manager that a new Stock Keeping Unit (SKU) has been created for the
product.</FreeFormText>

                </description>

             </ActionIdentity>

             <GlobalDocumentFunctionCode>Request</GlobalDocumentFunctionCode>

             <PartnerRoute>

                <fromPartner>

                   <PartnerDescription>

                      <GlobalPartnerClassificationCode>Distributor
                          </GlobalPartnerClassificationCode>

                      <BusinessDescription>

                         <GlobalBusinessIdentifier>123456789
                             </GlobalBusinessIdentifier>

                      </BusinessDescription>

                   </PartnerDescription>

                </fromPartner>

                <toPartner>

                   <PartnerDescription>

                      <GlobalPartnerClassificationCode>Manufacturer
                          </GlobalPartnerClassificationCode>

                      <BusinessDescription>

                         <GlobalBusinessIdentifier>987654321
                             </GlobalBusinessIdentifier>

                      </BusinessDescription>

                   </PartnerDescription>

                </toPartner>

             </PartnerRoute>

          </ActionControl>

       </TransactionControl>

   </ProcessControl>

</ServiceHeader>
```

```
     --RN-part-boundary
     Content-Type:  Application/XML; RNSubType="service-content"
     Content-Description: This is the Service Content part of the business
        message

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE Pip2A8ProductSKUCreationNotification SYSTEM
"Pip2A8ProductSKUCreationNotificationGuideline.dtd">

<Pip2A8ProductSKUCreationNotification>

   <ProductNotice>

          <theNotice><FreeFormText>Produt SKU created.
              </FreeFormText></theNotice>

          <GlobalProductIdentifier>00123456789012</GlobalProductIdentifier>


   </ProductNotice>

            <fromRole>

               <PartnerRoleDescription>

                  <GlobalPartnerRoleClassificationCode>Buyer
                      </GlobalPartnerRoleClassificationCode>

                  <PartnerDescription>

                     <GlobalPartnerClassificationCode>Distributor
                         </GlobalPartnerClassificationCode>

                     <BusinessDescription>

                        <GlobalBusinessIdentifier>123456789
                            </GlobalBusinessIdentifier>

                        <GlobalSupplyChainCode>Information Technology
                            </GlobalSupplyChainCode>

                     </BusinessDescription>

                  </PartnerDescription>

                  <ContactInformation>

                     <contactName><FreeFormText>A. Name</FreeFormText>
                        </contactName>

                     <telephoneNumber><CommunicationsNumber>299 688-8998
                        </CommunicationsNumber></telephoneNumber>

                     <EmailAddress>email@mail.com</EmailAddress>

                  </ContactInformation>

               </PartnerRoleDescription>

            </fromRole>

            <toRole>
```

```
            <PartnerRoleDescription>

               <GlobalPartnerRoleClassificationCode>Customer Manager
                   </GlobalPartnerRoleClassificationCode>

               <PartnerDescription>

                  <GlobalPartnerClassificationCode>Manufacturer
                      </GlobalPartnerClassificationCode>

                  <BusinessDescription>

                     <GlobalBusinessIdentifier>987654321
                         </GlobalBusinessIdentifier>

                     <GlobalSupplyChainCode>Information Technology
                         </GlobalSupplyChainCode>

                  </BusinessDescription>

               </PartnerDescription>

            </PartnerRoleDescription>

         </toRole>

         <thisDocumentGenerationDateTime><DateTimeStamp>
              19990530T132000.0500Z
              </DateTimeStamp></thisDocumentGenerationDateTime>

         <thisDocumentIdentifier><ProprietaryDocumentIdentifier>
            DOC-0001</ProprietaryDocumentIdentifier>
            </thisDocumentIdentifier>

         <GlobalDocumentFunctionCode>Request</GlobalDocumentFunctionCode>

</Pip2A8ProductSKUCreationNotification>
     --RN-part-boundary–
```

Bibliography

RosettaNet Documents

All documents listed in this section are published by RosettaNet. The latest versions can be found at the RosettaNet website (http://www.rosettanet.org).

- *PIP2A8: Distribute Product Stock Keeping Unit (SKU)*
- *Preamble Part Message Guideline*
- *Service Header Part Message Guideline*
- *Partner Interface Process Technical Architecture.*

Other Documents

Documents in this section are published by various organizations. Sources for the documents are indicated when known.

- *Open Buying on the Internet (OBI)™ Standard*, version 1.0, published May 1997 by The OBI Consortium. This document was based on the work of the Internet Purchasing Roundtable. Source: http://www.openbuy.org.
- *Open Buying on the Internet (OBI)™ Technical Specifications*, version 1.1, published June 1998 by The OBI Consortium. Source: http://www.openbuy.org.
- RFC 1521: "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies." N. Borenstein, N. Freed. IETF, Network Working Group, 1993. (Source: http://www.ietf.org/rfc/rfc1521.txt)
- RFC 2119: "Key Words for Use in RFCs to Indicate Requirement Levels." S. Bradner, Harvard University. IETF, Network Working Group, 1997. (Source: http://www.ietf.org/rfc/rfc2119.txt)
- *PKCS #1: RSA Cryptography Specifications*, version 2. B. Kaliski and J. Staddon, RSA Labs. The Internet Society, © 1998. (Source: http://www.rsa.com)
- *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)
- *PKCS #9: Selected Attribute Types*. An RSA Laboratories Technical Note. Version 1.1. Revised November 1, 1993. (Source: http://www.rsa.com)
- *S/MIME Implementation Guide,* Version 2. Steve Dusse, RSA Labs, ©1996. (Source: http://www.rsa.com)
- Recommendation X.200 (07/94) - Information technology – Open Systems Interconnection -- Basic reference model: The basic model. I.T.U. (Available at: http://www.itu.int/itudoc/itu-t/rec/x/x200-499/x200_25094.htm)
- 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)

GLOSSARY

ACID: A set of software tests (atomicity, consistency, isolation and durability) that are used to ensure hardware and software stability. For example, in transaction processing, a transaction has atomicity if the operations that make up the transaction either all execute to completion, or can be made to appear never to have occurred at all. A transaction has consistency when it successfully transforms the system and the database from one valid state to another. A transaction has isolation if it is processed concurrently with other transactions and still behaves as if it were the only transaction executing in the system. (It does not interfere with other transactions and others do not interfere with it.) A transaction has durability if all the changes that it makes to the database become permanent when the transaction is committed. (from Jonar C. Nader, Prentice Hall's *Illustrated Dictionary of Computing*, 3rd edition, 1998)

agent: An agent is a network component that must implement protocols up to the agent layer of the e-business network application communications model. An agent has no network identity as a business service component. A user agent acts as an intermediary between a business service and an employee. *See also* Business Service.

architecture: the way components of a complex system fit together (adapted from the *Free On-Line Dictionary of Computing*, wombat.doc.ic.ac.uk)

"as-is" Business Process Model: A graphical model of an organization'ss business process showing the activities, external processes and decisions along with the information exchanged between them. An employee fulfilling a role performs each activity. External processes are performed by external entities.

business service: A business service is a network component that responds to business transaction requests initiated by other services. A business service implements protocols in all of the layers of the e-business network application communication model. Services monitor the execution of execution processes. A service component has network identity as a business service. "Business service" has been shortened to "service" in most of this document. *See also* agent.

compliance: an implementation is *compliant* if and only if it fully meets each and every requirement of the RNIF specification, i.e., this document. 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)

control information: Information that is prepended and appended to content from an upper network layer.

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).

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)

extension: added functionality or features that do not violate the integrity of the original framework

framework: a set of related architectural components.

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

header: Control information prepended to content from an upper network layer.

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

interchange format: a specific data layout that defines a structured business document. The interchange format specifies the sequence, representation and grouping of granular data elements, and may describe each element in terms of data type, optionality, cardinality, size and valid values.

lower layer: A layer lower down in the OSI network communications reference model.

network layer (layer): OSI communications reference model comprises layers, each specifying particular network functions. Many protocols can implement the functions defined for a layer.

OBI object: the standard data structure used to exchange order-related data between OBI-compliant trading partners. The OBI object contains an encapsulated version of OBI data and it may include a digital signature. The OBI data field within an OBI object contains an order or order request that has been formatted based on the EDI-based OBI order format specification. (from Open Buying on the Internet Technical Specifications, Release V1.1)

Partner Interface Process (PIP): A model that depicts the activities, decisions and partner Role Interactions that fulfill a business transaction between two partners in the IT 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.

Partner Role Interaction: The one-way exchange of business properties between partner roles. Partner Role Interactions are executed by networked service and agent applications that interact by transmitting and receiving messages. These services interact via service transactions and the agents with services. Both interact by transmitting and receiving messages.

PIP: *See* Partner Interface Process (PIP)

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

RosettaNet Object: the RosettaNet version of the "OBI Object" (see above), which accommodates upper level protocols.

segment: a group of logically related data elements in a defined sequence.

service: *See* business service.

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

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

"to-be" Business Process Model: A graphical model of an organization's business process, functioning as a partner type in the IT supply chain, showing the activities, external partner processes and decisions along with the properties exchanged between them. An employee fulfilling a role performs each activity. External processes are performed by IT supply chain partner types. Property exchange between IT supply chain partners is depicted as a Role Interaction.

trailer: Control information that is appended to content from an upper network layer.

transaction: A transaction is a collection of actions that have ACID properties. ACID is an acronym or Atomicity, Consistency, Isolation and Durability. Atomicity means that a transaction is an indivisible unit of work. Consistency means that a completed transaction must leave the system in a correct state or it must abort. Isolation means that a transaction behavior is not affected by other transactions that execute concurrently. Durability means that transaction' effects are permanent after it commits.

transaction set: A collection of formatted data that contain the information required by a receiver to perform a standard business transaction.

upper layer: A layer higher up in the OSI network communications reference model.

user: a person who requests RosettaNet services via a web browser.

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, i.e., this document, 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 *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.