Lund University
Department of Communication systems

The RosettaNet Standard and Compliant Platforms

A study of the RosettaNet Business-To-Business communication standard and its role in business and technical integration platforms.

Albin Kjellin

Abstract

Modern technology and the Internet in particular have revolutionized the way business is conducted. The need is extensive for collaborations between and amongst business and information systems. Electronically based trading has proven to provide a number of advantages such as increased speed, reliability and efficiency. The term “business process” has been extended to include collaboration between trading partners apposed to just describing the internal information flow. To be able to integrate organizations IT systems with each other, a global framework for e-business has to be established. There have been many initiatives from global organizations and the latest one is the RosettaNet standard.

The most central conception of the RosettaNet standard is the Partner Interface Process (PIP) and it describes the choreography of a business message exchange between two business partners. The PIP:s are in turn controlled by the RosettaNet Implementation Framework (RNIF) that specifies the more grammatical aspects of the business process. The context of a business message is specified in two dictionaries, one technical and one business oriented. The simplest way to describe the RosettaNet standard is to look at it as a language where the dictionaries provides the words, the RNIF the grammar and the PIP:s describe the dialog.

The purpose of this project is to examine two RosettaNet compliant platforms, IBM WebSphere Partner Gateway and BEA WebLogic, to determine pros and cons in terms of validation level, complexity and connectivity options. The testing of these two platforms has been conducted using a test client provided by the RosettaNet consortium. The client application is called RosettaNet Self Test Kit (RNSTK) and is used to validate compliance to the RosettaNet standard in terms of structure and syntax. Both WebLogic and WebSphere Integration Connect have been tested using this application. They have also been tested against each other to determine the level of validation. IBM WebSphere Integration Connect is by far the application that has the most complete support for the RosettaNet standard. Messages that pass the validation in both RNSTK and WebLogic generate an exception in WebSphere Integration Connect.

One part of this project was to modify the RNSTK to provide extra functionality. After the modification the RNSTK can receive any type of RosettaNet message and provide a response without needing prior configuration for that particular test scenario.

Sammanfattning

Modern teknik och Internet i synnerhet har revolutionerat sättet på vilket affärstransaktioner utförs. Behovet av integration mellan affärs- och informations system är enormt och kommer inte att minska med tiden. Elektronisk handel har visat sig ha många och betydande fördelar som snabbare processer, ökad tillförlitlighet och effektivitet jämfört med dess manuella föregångare. Förut användes bara termen affärsprocess inom en organisation men när man talar om affärsprocesser idag innefattar det även transaktioner mellan skilda organisationer. För att på allvar kunna ta del av alla fördelar en väl integrerad affärsprocess erbjuder måste det finnas standardiserade sätt för hur den sådan affärsprocess är uppbyggd. Det har på senare tid dykt upp ett antal initiativ från olika branschorganisationer för att skapa en global standard. En av dessa och den förmodligen mest populära är RosettaNet standarden.

Det mest centrala begreppet i RosettaNet-standarden är Partner Interface Process (PIP) och det beskriver flödet av affärsmeddelanden mellan två partners. PIP processer specificeras i sin tur av något som kallas RosettaNet Implementation Framework (RNIF) som definierar de mer grammatiska aspekterna av en affärsprocess. Innehållet i ett affärsmeddelande specificeras i två ordböcker, en innehåller affärstermer, den andra tekniska termer. Det enklaste sättet att beskriva RosettaNet standarden är att se på den som ett språk där ordböckerna beskriver ordern, RNIF grammatiken och PIP processerna dialogen.

Syftet med detta projekt var att undersöka två stycken platformar med RosettaNet-stöd, IBM Partner Gateway och BEA WebLogic, för att fastställa för och nackdelar vad det gäller validerings nivåer, komplexitet och alternativ för integration med andra produkter. Tester av dess två produkter har utförts med hjälp av en testklient, tillhandahållen av RosettaNet organisationen. Klientapplikationen kallas för RosettaNet Self Test Kit (RNSTK) och används för att validera syntax och struktur av RosettaNet meddelanden. Både IBM WebSphere Integration Connect och BEA WebLogic Integration Platform har testats mot denna applikation. Detta kombinerat med tester mot varandra har använts för att fastställa nivåer av validering hos de olika produkterna. Resultatet av dessa tester visare att WebSphere Integration Connect har den absolut mest stringenta valideringsprocessen. Medelanden som passerar både RNSTK och BEA WebLogic Integration generar fel i WebSphere Integration Connect.

En annan del av projektet var att modifiera RNSTK för att tillföra funktionalitet. Innan modifikationen var man tvungen att konfigurera klienten och ställa in vilken PIP den skulle delta i. Efter modifikationen är det möjligt för klienten att svara på en godtycklig PIP förfrågan.

Preface

This project was performed at Connecta AB locations in Stockholm. The master thesis is the final examination for my master degree in electrical engineering at Lund University. Responsible department at Lund Institute of Technology is Communication Systems.

This project has been a great experience for me and it has been interesting from day one. It has taught me a lot, not just from a technical aspect, but I have also learned how it can be to work in an environment filled with highly competent instructors and “co-workers”. I hope that this report to some extent can pass on the learning I gained during the course of this project and hopefully inspire others to go on where I left of.

I would like to thank my instructor at Connecta AB, Jonas Kvist for all his efforts to help me in day-to-day work and providing an outline for the project.

I would also like to thank Johan Henriksson at Connecta AB, for providing the visions for this project as well as for help with small technical issues.

From Lund University of Technology I would like to thank Per Runeson for help with the report and project planning.

As Connecta AB provided has provided the tools needed for this project I also have to direct my gratitude to the company. Connecta AB has loaned me use to a laptop, access to a server, a desk and all needed software.

Table of contents

1 Project Scope

1.1 Objective

The overall goal of this project is to find a way to configure a system or establish an outline for a new implementation that can be used in deployment of a business-to-business communication solution that implements and verifies compliance to the RosettaNet standard. This will be achieved by an inventory examination of available platforms, examining which configurations that can be made and to what level they can implement and validate the RosettaNet standard. Another aspect of the project is to determine ways of efficient procedures for how to test and validate RosettaNet solutions.

The applications will be configured so that business messages can be sent between them to determine pros and cons in terms of complexity, functionality and level of validation. Each platform will be examined in terms of connectivity with other business applications such as SAP R3.

1.2 Approach

This project is mainly concentrated on investigation and evaluation of existing systems and solutions. The initial part of the project is focused on achieving an understanding of the technical and business context of RosettaNet. Further on a theoretical study of the technical elements, such as XML Schemas and validation methods, is conducted.

The actual testing is done with a hands-on approach where systems are installed, configured and tested against each other to determine their characteristics. These tests will be conducted on deployed systems from different vendors such as BEA WebLogic and IBM WebSphere.

Results and experiences gained from testing and literature studies are documented in a structured fashion in the discussion part of the report. Similar projects and research reports are studied and compared to the conclusions and experiences from this project.

1.3 Scope

The scope of this project is restricted to give a theoretical background to RosettaNet, related technologies and an investigation of current applications and methods for the RosettaNet standard. The following platforms are examined, IBM Websphere Integration and BEA WebLogic Integration. The platforms are examined in terms of efficiency, availability and functionality. The decision to focus on these two platforms is based on availability of software and existing competence at Connecta AB.

The possibility for extensions of the test client supplied by the RosettaNet consortium is examined. The test application is supposed to work as an aid to developers when implementing and validating new RosettaNet solutions.

Other issues like RosettaNet adapters for different backend systems and compliance with company specific terminology are not be of primary focus in this report. But a general discussion of how to integrate RosettaNet compliant applications with standard business landscape architectures will be presented.

1.4 Stakeholders

There are two main stakeholders in this project and that is Connecta AB and Lund University. The project is conducted at Connecta AB:s premises in Stockholm. Connecta AB also supplies all necessary equipment for the project such as workstation and servers. All results will be presented to LTH in form of this report and an oral presentation.

Connecta is an IT/Management consultant company located in Stockholm. They have about 300 employees. Connecta AB has four main areas of expertise, Development, Enterprise Applications, Management and Technology.

1.5 Intended Audience

This report is foremost intended for an audience with a technical background although it has its origin in a business context. Some basic knowledge in computer science is preferred.

2 Introduction

2.1 Background

2.1.1 Business Context

The world is getting smaller and trade is conducted in a global environment for most types of consumer gods. As consumers we are getting more and more restless and we demand new and cheaper products. This in turn forces manufacturers and logistic companies to streamline supplier, production and transportation chains. To be able to meet the demands on reduction of production costs, a lot of manufacturing has been outsourced to developing countries such as China and India. The outsourcing process has in many cases led to that cultural difference become a source for misunderstandings and confusion. To overcome this barrier the need for a strict standard for B2B communication has been substantial. The use of electronic B2B communication is widespread among larger organizations all over the world and many supply chains are more or less dependent on IT infrastructure for trade and information exchange. This is a fast growing phenomenon according to Marin Schoeppler from Agilent Semicondutors. In his lecture Supply Chain Integration [1] he claims that 30 % of B2B communications was done using Electronic Data Interchange compared to 70 % that was conducted using traditional means like fax or telephone. In 2002 the use of EDI was 40 % and traditional means were 60 %. In 2002 the estimations for 2005 the allotment was 60 % for the RosettaNet standard, 30 % using EDI and merely 10 % using telephone or fax. The above numbers are taken from an article in Elektronik i Norden. According to C C Albrecht et al [2] the dollar volume of e-commerce in general was 700 billion USD for 2001 and was expected to be 8.5 trillion USD for 2005, of which 90 % will be B2B commerce.

2.1.2 EDI

The Electronic Data Interchange (EDI) is a standard for B2B communication used by a number of large organizations. It is a specification for how two business systems should communicate directly with each other. Depending on specific versions and company specific implementation, it can be used for transactions like invoicing, quotation, and consignment notes (a more detailed explanation of technical aspects of EDI is presented in section 3.1.1 in this report). EDI was founded in 1970 and is still widely used today, though the common opinion is that this standard will be abandoned in the near future. One reason for this is that the standard is so stringent that many companies have been forced to customize the protocol to fit their demands. This has led to that the original standard has mutated into a large number of sub standards. Because of this mutation the implementation of EDI's has become very expensive since every new business partner require, if not a completely new implementation then at least an extensive customisation of the current B2B communication system. All of these prerequisites have led the industry to look into new and more modern solutions [3]. A similar view is presented by CC Albrecht et al [2] who mention the lack of standardisation for product searching and support for engaging new participants as a major drawback for EDI.

As the Internet has grown to be the global platform that it is today, a logic conclusion is that the standard will be constructed with this in mind. Most businesses all over the world already have access to the Internet and is more or less dependent of it in their daily work. There are a few ongoing initiatives from different organizations to establish new standards more suited for business transactions via the Internet. There are two standards that have been more recognized than others, namely ebXML and RosettaNet. Both these standards are XML-based.

2.1.3 ebXML

ebXML stands for Electronic Business using eXtensible Markup Language and is governed by the Organization for the Advancement of Structured Information Standards (OASIS) and CEFACT (United Nations/ECE agency). ebXML was created with the ambition to be suited for many different types of organizations, hence it is more generally applicable and complex then for example the RosettaNet standard that has a clear focus on certain industries. There are a few case studies conducted at organizations that have chosen to implement a B2B solution using an ebXML. Most of them mention a reduction in cost for manual work and validation of business documents. Another cost saver is the reduction of paper costs. One of the largest improvements is the possibility to save all of the transaction history to a local database in a simple manner [4].

2.1.4 RosettaNet

RosettaNet was founded in the U.S. in February 1998, RosettaNet is an independent, non-profit consortium of more than 500 organizations. These organizations include some of the world's leading electronic components, computer and consumer electronics, semiconductor manufacturing, telecommunications, and logistics companies. The main goal for the RosettaNet standard is to improve speed, efficiency and reliability of B2B transactions to enable better conditions for communication and collaboration between trading partners. RosettaNet provides a common platform for trading partners who want to conduct business over the Internet. Since the standard is very formal it gives companies the ability to automate their business processes to a much higher level than before. The main difference between RosettaNet and EDI is that EDI exchanges documents between organizations while RosettaNet defines the whole business processes and the network is a natural part of the transaction. There has been many studies conducted on RosettaNet and EDI and the most common opinion is that the following are the most significant advantages of RosettaNet compared to EDI [5].

- An easier and more cost efficient implementation, with a greater return on investment (ROI).
- The ability to automate a larger number of business processes.
- Real-time transaction handling as opposed to batch processing.
- Greater scalability.

2.1.5 Intel vs Shinko Case Study

The following example is taken from the RosettaNet homepage and it is a ROI case study of a RosettaNet project involving Intel and Shinko. The business challenge of this project was to replace the current manual procurement system and establish a 100 % electronic forecast-to-cash procurement process. Before the RosettaNet implementation the processes included a lot of manual work. Everything from typing in values for a forecast to creating an invoice where done manually. Then these business messages where sent by fax or email.

Both companies had prior experience of RosettaNet and managed to complete the implementation only using in-house IT capability. The RosettaNet implementation did not demand significant changes to the backend business system, but since many steps of the processes where made to assist the following manual work, they could be removed to streamline the processes. When the project was completed the different business processes where totally automated and there was no need to use email or fax.

The fact that all manual work was eliminated in turn has contributed to a major efficiency gain at Shinko. The following list shows the efficiency gain measured in annual workload savings at Shinko after completing the project with Intel [6]:

- Inventory management 86% or 650 hours.
- Forecast process 88% or 416 hours.
- Invoice and payment 65% or 286 hours.

When reviewing these figures one has to take in to consideration that the efficiency gain was compared to a total manual process. For most larger corporations this is not the case and a their business processes may be automated using network exchange to some extent.
There is a number of other case studies available at the RosettaNet website and they all show an improvement in efficiency [6].

3 Technical Context

3.1 EDI

During the 1970s a new standard for B2B communication, Electronic Data Interchange (EDI) was developed by the car and transport industry. At first only to handle order and invoicing. The standard was a stringent framework of specifications that forced the users to configure their systems accordingly. In the latest versions of EDI the standard has been more adjusted to Internet and supports the XML standard. There are basically two ways to adapt to an EDI standard. Many larger organizations choose to implement their own version of the EDI and smaller organizations, depending on these larger organizations, have no choice then but to adapt to that EDI version. Another way is to connect to a Value Added Network (VAN) that supplies its subscribers with transaction service, security, standard formats and communication protocols. Most VAN:s also supply the communication network.

There are basically three different standards of EDI [7]:

- ANSI ASC X12 (American National Standards Institute): This standard is mostly used in USA. The standard defines a large number of documents such as consignment notes, loan applications, damage reports and invoices.
- EDIFACT (Electronic Document Interchange for Administration, Commerce and Transportation): This is the international standard and it is governed by the United Nations Trade Data Interchange Directory (UNTDID). The standard holds syntax rules, implementation guides, file structures and data elements. It is based on a combination of X12 and Trade Data Interchange (TDI) that is the European standard.
- Open-EDI: This standard is the result from a collaboration between the International Standard Organization (ISO) and the International Electrical Committee (IEC). The purpose of this standard is that it should have a better availability then the other standards. It is made to work between two organizations with no prior business relation.

3.2 ebXML

ebXML are an XML based standard that is used for business-to-business communication. Below is a short description of the different components of the standard [4].

- Business Process Specification Schema (BPSS) is a schema that describes the dialog of messages that is sent between business partners. It is available as a DTD and a XSD.
- Collaboration Protocol Profile (CPP) describes technical capabilities and supported business collaboration types of an organization. CPP:s can include BPSS:s.
- Collaboration Profile Agreement (CPA) is an agreement of business collaborations and technical capabilities two organizations have agreed to use in mutual communication, these must be supported by CPPs of both organizations.
- Registry Service (ebRS) is storage for information needed to conduct interactions with other trading partners. The ebRS can hold different CPP:s.
- Registry Information Model (ebRIM) specifies the information model employed in the registry service
- Messaging Service (ebMS) specifies information about the actual messaging such as encryption and security.

3.3 XML Standards

3.3.1 XML

XML stands for eXtensive markup Language and is a way to save and structure data. A problem today is that almost all applications use different formats to present and transfer data. The purpose of XML is to provide a more general way to describe data so that it will be more accessible. A common misperception is that XML will replace HTML. The main task for HTML is to present data and XML is only used to describe and structure data. Another difference is that the tags in HTML are predefined and in XML you have to define your own tags.

An XML document is built up by elements that contain textual data or other elements. The elements are used to build a document with a tree-like structure. Every XML element consisting of one or more sub elements is called a parent and a sub element to a parent element is called a child. Every XML document must have a "main parent" element called the root that contains all other elements [8].

3.3.2 DTD

DTD stands for Document Type Definitions and is a standard for how to define valid building blocks of an XML document. A DTD can be declared inside an XML document or in an external file. An XML file can be parsed against a DTD to validate the structure and compliance with the predefined tags. It is practical to use DTD:s when two applications are using XML documents to communicate with each other in a server-client based fashion. To guarantee that the data that is sent has the proper syntax and structure, the client may validate the XML document to be sent against a DTD to ensure that the server can handle the data. The same validation can be preformed at the server side of the application. A DTD specifies how many children a specific element can have and what their names should be. There is another technique for validating XML documents that is said to replace the DTD in the future. This technique is called XML Schema and is described below [8].

3.3.3 XML Schema Language

XML Schema Language is also defined as XML Schema Definition (XSD). As with the DTD the purpose of XML Schema is to define the legal building blocks of an XML document. XSD divide elements in to two basic categories, simple and complex. The simple elements can only contain text. Complex elements can contain attributes and/or other elements. There are four kinds of complex elements:

- Empty elements
- Elements that contain only other elements.
- Elements that contain only text.
- Elements that contain both other elements and text.

The XSD provide a great number of possibilities when it comes to validation. It is possible to define how many occurrences of a child element that is valid, which names can be used and in which sequence they should appear [8].

3.3.4 XPath

XPath is a language for finding information in an XML document. XPath is used to navigate through elements and attributes in an XML document. XPath divides an XML document in to seven kinds of nodes:

- Element
- Attribute
- Text
- Namespace
- Processing-instruction
- Comment
- Root node

This distinction of nodes is used when a query for a certain part or parts of an XML file is wanted. It is possible to acquire information from the XML file with certain distinct properties. For example if an XML-file is describing a phonebook it is possible to acquire all sir names from the file using an XPath expression [8].

3.4 RosettaNet

The RosettaNet standard forms a e-business language for communication between trading partners. The name RosettaNet originate from the Rosetta stone found in Egypt in 1799 and was dated back to 196 B.C. The stone had inscriptions of one message in two different language using three different scripts. This helped the archaeologist to eventually decipher the hieroglyphics. This is in line with the main goal for the RosettaNet standard which is meant to work as a global language for business transactions overcoming the barriers of communication between different business systems. Today there is about 100 PIP:s defined. The PIP:s are at an implementation level represented by DTD and XSD files, this improves the possibilities for building and validation using standard XML techniques [5].

Figure 1 shows the three different levels of the RosettaNet standard. A short introduction to the building blocks are presented here and then a more detailed description of the parts follows.

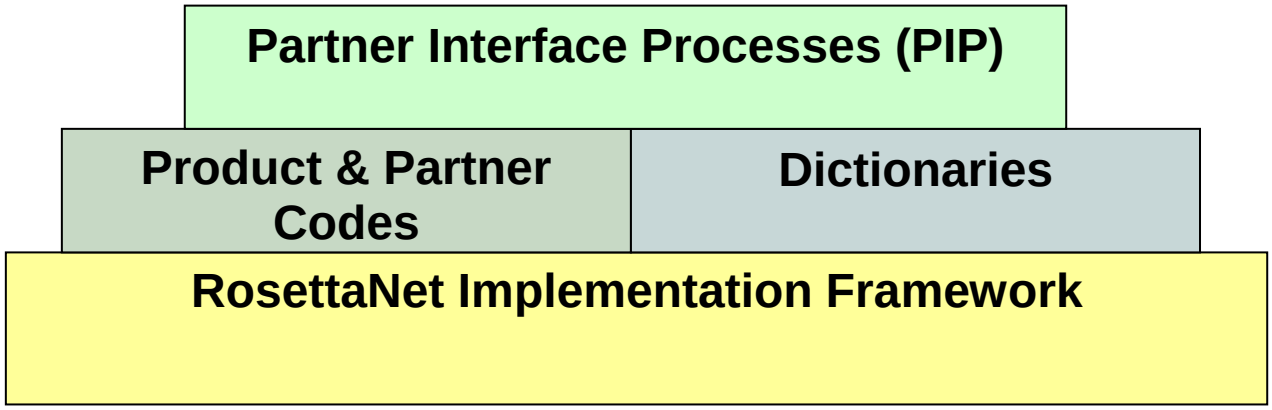

Figure 1: Overview of the RosettaNet Standard.

- **Partner Interface Processes (PIP):** Describes the choreography of a message dialog between to trading partners and vocabulary for system-to-system communication.

- **Dictionaries:** Describes the words used in business documents. There is one Technical Dictionary (TD) and one Business Dictionary (BD).
- **RosettaNet Implementation Framework (RNIF):** Specifies the information exchange using XML in a B2B scenario in terms of transport, routing, packaging, security, signals and trading partner agreements.

Compared to a language the dictionaries would be equivalent to the words the RNIF would be the grammar and the PIP:s would correspond to the dialog.

3.4.1 PIP

The PIP architecture is divided in to five different levels, Clusters, Segments, PIP:s, Activities and Actions. In these levels different aspects of the PIP is described.
According to the RosettaNet consortium there is four criteria's that a PIP has to posses [10]:

- Provide a measurable business outcome or output.
- Contain non-proprietary business processes.
- Include more than one role interaction.
- Stand as discrete units of work that can be attached and built into other PIP:s to achieve a larger business outcome.

The PIP:s are divided in to eight clusters each containing a different unit of B2B communication:

- **Cluster 0:** RosettaNet Support.

  Holds administrative functionality.

- **Cluster 1:** Partner Product and Service Review.

  Holds PIP:s for information gathering, maintenance, distribution for development of new business partners profiles product information subscriptions.

- **Cluster 2:** Product Information.

  Holds PIP:s for distribution and periodic updates of product and design information, including product change notices and detailed technical specifications.

- **Cluster 3:** Order Management.

  Supports the full order-management business area, from price and delivery quoting through purchase order initiation, status reporting, and management. Order invoicing, payment, and discrepancy notification are also managed using this cluster of processes.

- **Cluster 4:** Inventory Management.

Enables inventory management, including collaboration, replenishment, price protection, reporting, and allocation of constrained products.

- **Cluster 5:** Marketing Information Management.

  Enables communication of marketing information, including campaign plans, lead information, and design registration.

- **Cluster 6:** Service and Support.

  Provides post-sales technical support, service warranty support, and asset management capabilities.

- **Cluster 7:** Manufacturing

  Enables the exchange of design, configuration, process, quality, and other manufacturing floor information to support a "virtual manufacturing" environment.

Each cluster is then divided into two or more segments. Each segment holds PIP:s with similar functionality. As an example cluster three has four segments.

Cluster 3: Order Mangement
Segment A: Quote and Order Entry
Segment B: Transportation and Distribution
Segment C: Returns and Finance
Segment D: Product Configuration

The next level is the actual PIP:s that in turn is divided in to activities and action. The PIP:s define the dialog in a server/client based manner. The messages exchanged between the buyer and seller is divided in to activities and actions. Below is an example of how everything is structured from cluster to action level.

Cluster 3: Order Mangement
Segment A: Quote and Order Entry
PIP 3A1: Request Quote
Activity: Request Quote
Action: Quote Request Action
Action: Quote Confirmation
Segment B: Transportation and Distribution
Segment C: Returns and Finance
Segment D: Product Configuration

The difference between an activity and an action is that the action is the business messages sent between the different trade partners. An activity defines the function and which actions are required in that function.

The documentation of PIP specifications is divided in to three main parts describing the different aspects of the PIP. These parts are referred to as Business Operational View (BOV), Functional Service View (FSV) and Implementation Framework View (IFV) [9].

3.4.2 Business Operational View

The Business Operational View (BOV) captures the semantics of business data entities and their flow of exchange between roles as they perform business activities. It holds information about purpose, state, partner roles, process activity controls, business data and a business process definition. The most informative part of the BOV is probably the business process flow diagram that illustrates process as a kind of state machine. The buyer and seller in the Figure 2 are referred to as partner roles [9].

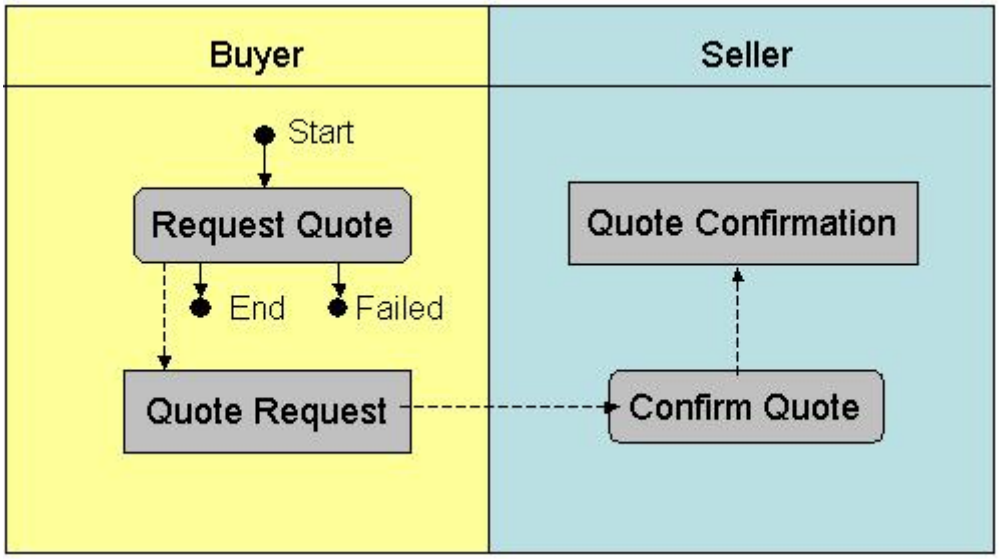

Figure 2: Business Operational View flow chart.

3.4.3 Functional Service View

Functional Service View (FSV) originates from a BOV and describes the interactions between the network component services in a PIP see Figure 3 and 4. It includes all transaction dialogs in the PIP and they are described by a network component design and a transaction dialog specification [9].

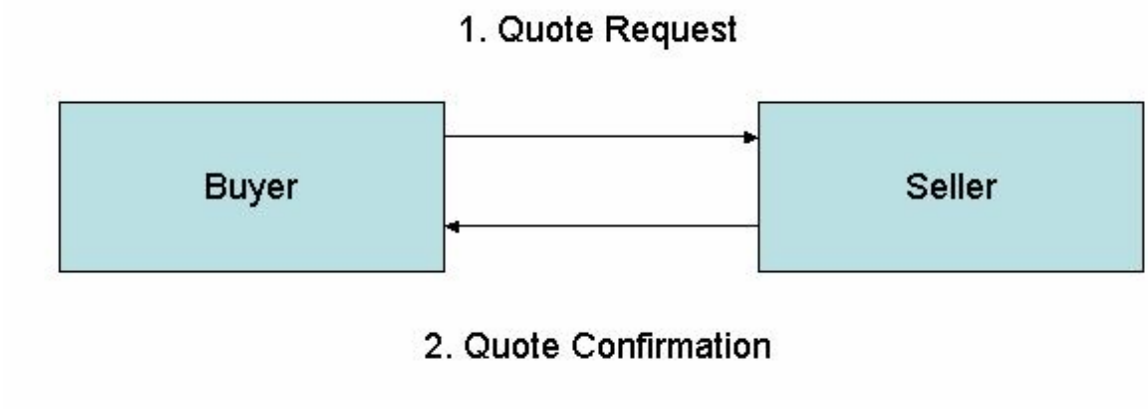

Figure 3: Functional Service View action chart.

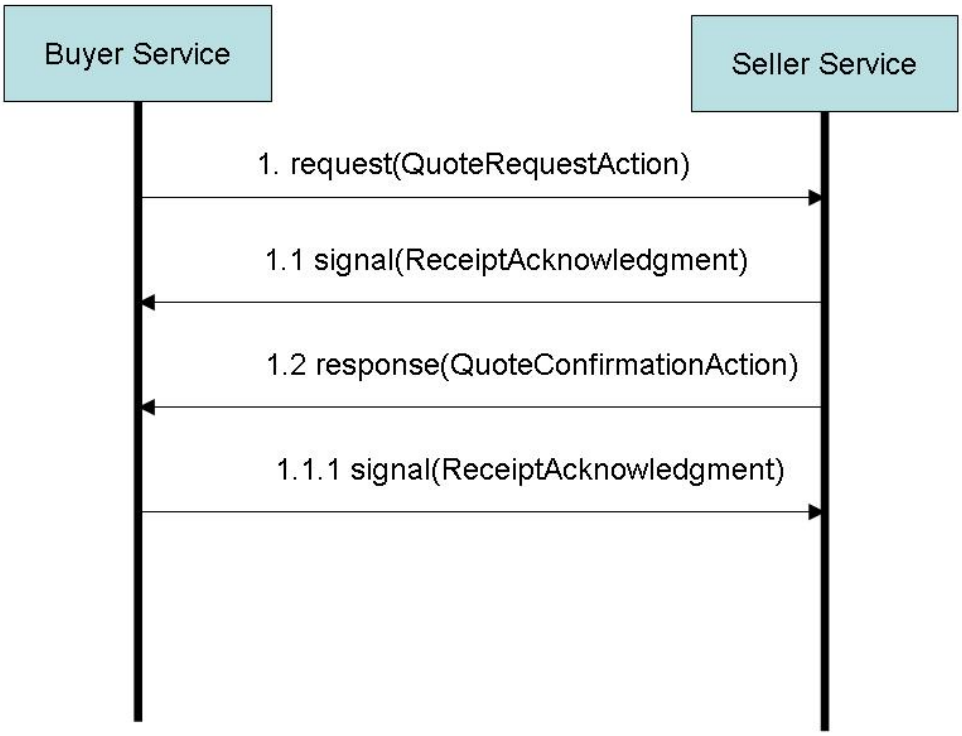

Figure 4: Functional Service View flow chart.

3.4.4 Implementation Framework View

The IFV specifies the action message formats and communication requirements between RosettaNet services according to the RosettaNet Implementation Framework (RNIF). It specifies if the business messages require a digital signal and if they need to be transported with a Secure Socket Layer (SSL) [9].

3.4.5 Dictionaries

A big source for confusion in B2B communication is the diversity of terminology amongst organizations. This has caused a lot of problems when earlier efforts to automate business processes have been made. To deal with this problem the RosettaNet organization has provided two different dictionaries to specify a valid vocabulary for e-business. There is one dictionary for business terminology (BD) and one for technical terminology (TD). Both the BD and TD are apart from the human readable version represented in XML and DTD files [10].

3.4.6 Business Dictionary

The RosettaNet Business Dictionary (RNBD) defines the following different business terminologies. Since this dictionary is supposed to cover most of the terminology used for well over a 100 business processes it is quite extensive. Below is a list of the entities that make up the RNBD [11]. There is a short example presented for each entity.

- Business Data Entities

  Example:

  Name: Discounts

Definition: The collection of business properties that describe payment discounts.

- Business Properties
  Example:
  Name: Discount Amount
  Definition: The financial amount representing a reduction to the total amount due.
- Entity Instances
  Example:
  Entity: Global Country Code
  Instance: SE
  Definition: Sweden
- Fundamental Business Data Entities
  Example:
  Name: Global Country Code
  Definition: Code identifying the two character country code specified in ISO 3166-1993.
  Type: String
  Min: 2
  Max: 2
  Representation: X(2)
- Quantitative Fundamental Business Data Entities
  Example:
  Name: Weight Dimension
  Definition: The weight of an artefact.
  Type: Real
  Min: 1
  Max: 15
  Representation: 9(13)V99

3.4.7 Technical Dictionary

The RosettaNet Technical Dictionary (RNTD) is far more extensive then the RNBD since it has to hold a lot more information. At first every industry had their own TD but now there is one central TD to cover all industry needs. The RNTD is designed to support unambiguous and automated electronic exchange of production information [10]. This is achieved by standardizing the semantics used to describe product characteristics and information. As an example, let's take a look at the definition of a photocopier:

```
<class id="RNIC021" propDefs="RNIS001 RNIS043 RNS-XJA001">
    <identifiers>
  <code>RNIC021</code>
  <majRev>001</majRev>
  <date.def>2000-12-05</date.def>
    </identifiers>
    <names>
  <preferred.name>COPIER</preferred.name>
    </names>
```

```
    <definition.short>A machine used to make photographic copies of
pages.</definition.short>
    <app.specific name="industry.domains">IT</app.specific>
</class>
Copyright © 1997 IEC, Geneva, Switzerland.www.iec.ch.
```

For each component it is defined which class it belongs to and how its properties are measured and by which unit of measurement [12].

3.4.8 Product and Partner Codes

RosettaNet product and partner codes are one of the efforts to align business processes between trading partners. The dictionaries are created to act in conformity with the product and partner codes. The following definitions are taken from the RosettaNet homepage [9]:

> *Global Company Identifier* — RosettaNet specifies the Data Universal Numbering System (D-U-N-S®) for Global Company Identifier in its PIP:s. The nine-digit D-U-N-S Number is a worldwide standard for company identification, distinguishing unique business locations around the globe [13].
>
> *Global Product Identifier* — RosettaNet specifies the Global Trade Item Number (GTIN) for Global Product Identifier in its PIP:s. The GTIN is a worldwide multi-industry standard for trade-item identification. GTINs are 14-digit numbers that uniquely and globally identify products and services [14].
>
> *Global Class Identifier* — RosettaNet specifies the United Nations/Standard Product and Services Code (UN/SPSC) for Global Class Identifier in its PIP:s. The UN/SPSC is an open, global commodity code standard for classifying products and services. Items are classified using numbers derived from the system's five-level hierarchy in which two digits are assigned at each level [15].

3.4.9 RosettaNet Implementation Framework

Next to the PIP:s the RosettaNet Implementation Framework (RNIF) is the most important part of the RosettaNet standard. The RNIF is a central document for organizations that plan to implement a RosettaNet solution. The main part of the specification is divided in to six sub chapters that describe how to structure the different elements of implementation. RosettaNet makes use of a number of other standards like MIME and HTTP and the relations between these standards are described. The following section describes the different aspects of the RNIF 2.0 [10].

3.4.10 RosettaNet Business Message Components

This section describes how RosettaNet makes use of the XML standard and how business messages are to be validated using standard XML validation techniques. It also describes the structure of the business messages in terms of headers, format specifications and payload components. A basic RosettaNet Business Message contains the following overhead units. These definitions are taken from the RNIF Core Specification 02.00.00.

- Preamble: This header identifies the standard with which this message structure is compliant.

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

These headers are used by the recipient to determine that it is a RosettaNet business message and what version of the RNIF that is used, the context of the message and to identify the sender for authentication and authorization. The structure and semantics of these headers are also described by and validated against DTD's. The remaining part of the business message is referred to as the payload containing the actual PIP action and possible attachments. In the RNIF these parts are referred to as Service Content and Attachments. The Service Content contains the actual business information in XML specified by the DTD for that particular PIP. Attachments can be any file type like a Word or PDF document, attachments are not mandatory in the business message but are presented as an option to complement the information stated in the Service Content. It is also possible to ship non-RosettaNet compliant service content in the payload but this is has stringent restrictions and must be agreed upon prior to execution by the trading partners [10].

3.4.11 Security Provision and Trading Partner Authentication

RosettaNet uses the "S/MIME Version 2 Message Specification" governed by the IETF RFC 2311 and this part of the RNIF describes the use of this standard. The S/MIME provide functionality both for enveloping of messages and digital certificates. MIME is the same technique used in common e-mail applications [9].

3.4.12 RosettaNet Business Message Packing and Unpacking

The Business message packing section of the RNIF describes how to pack and unpack messages that have been digitally sign and/or encrypted as well as plain messages. The following pictures illustrate the message structure in three different cases each using digital signature.

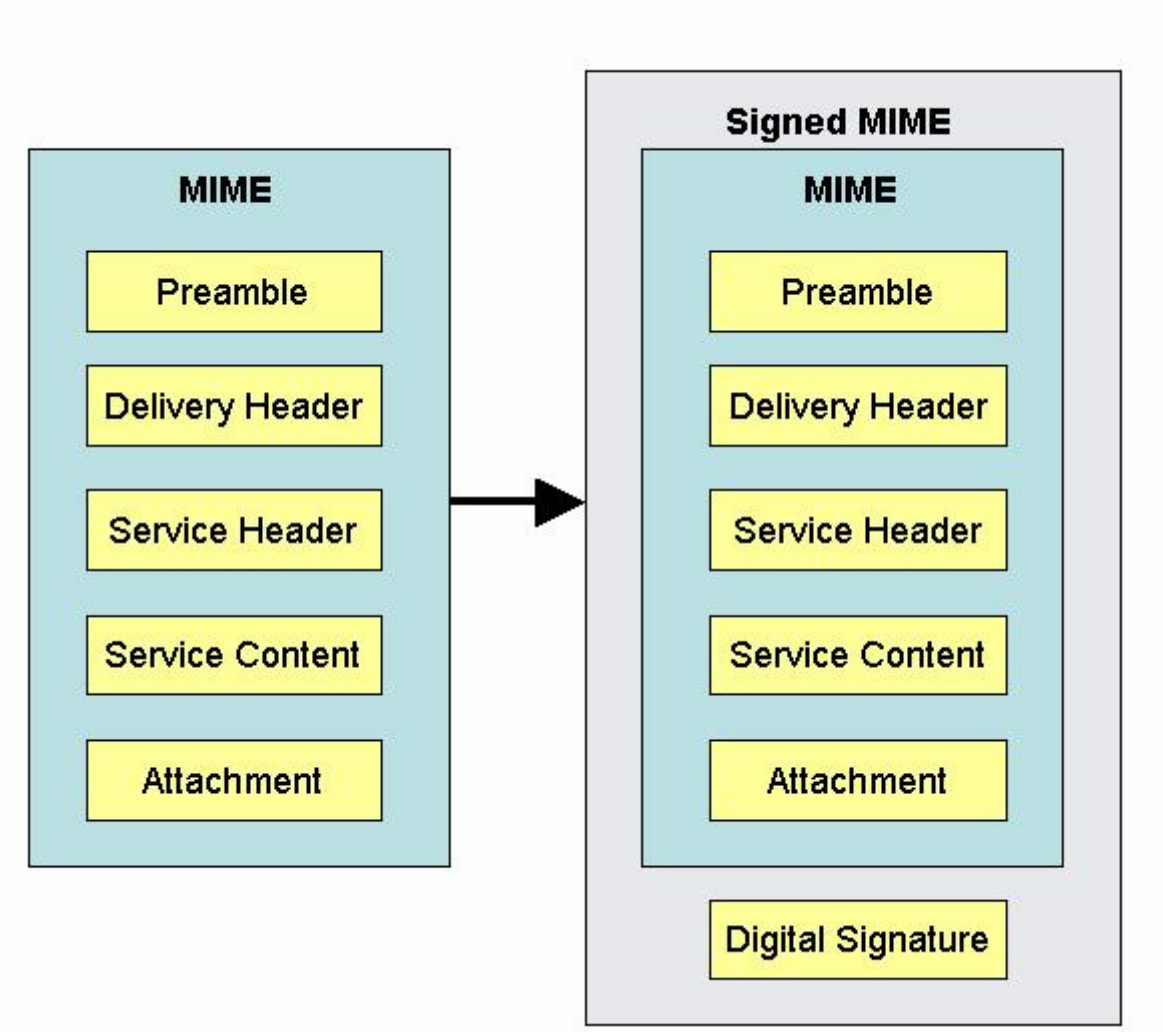

Figure 5: Signed RosettaNet message packaged in a MIME envelop.

The next simplest form after a basic RosettaNet XML message is a RN message packed in a signed MIME envelope see Figure 5. When a message is in this form all data are visible to the user. This can be compared to an e-mail that basically uses the same technique.

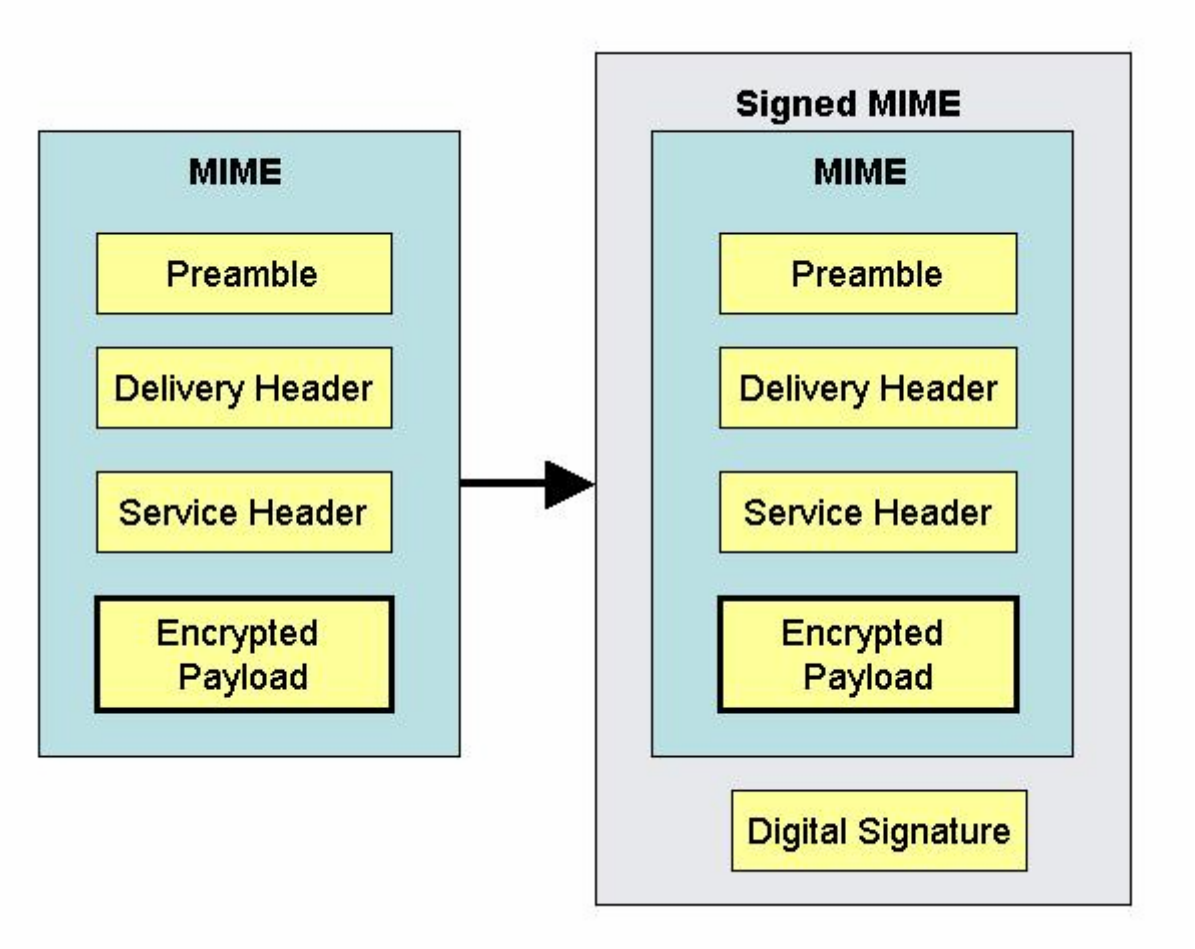

Figure 6: Signed RosettaNet message with encrypted payload packaged in a MIME envelop.

The next level of complexity is signed MIME packed messages with encrypted payload, see Figure 6. Payload in this context means the actual service content of the RN message along with attachments. This still leaves the Service Header visible to the public and it is possible to determine what kind of message is sent and to whom.

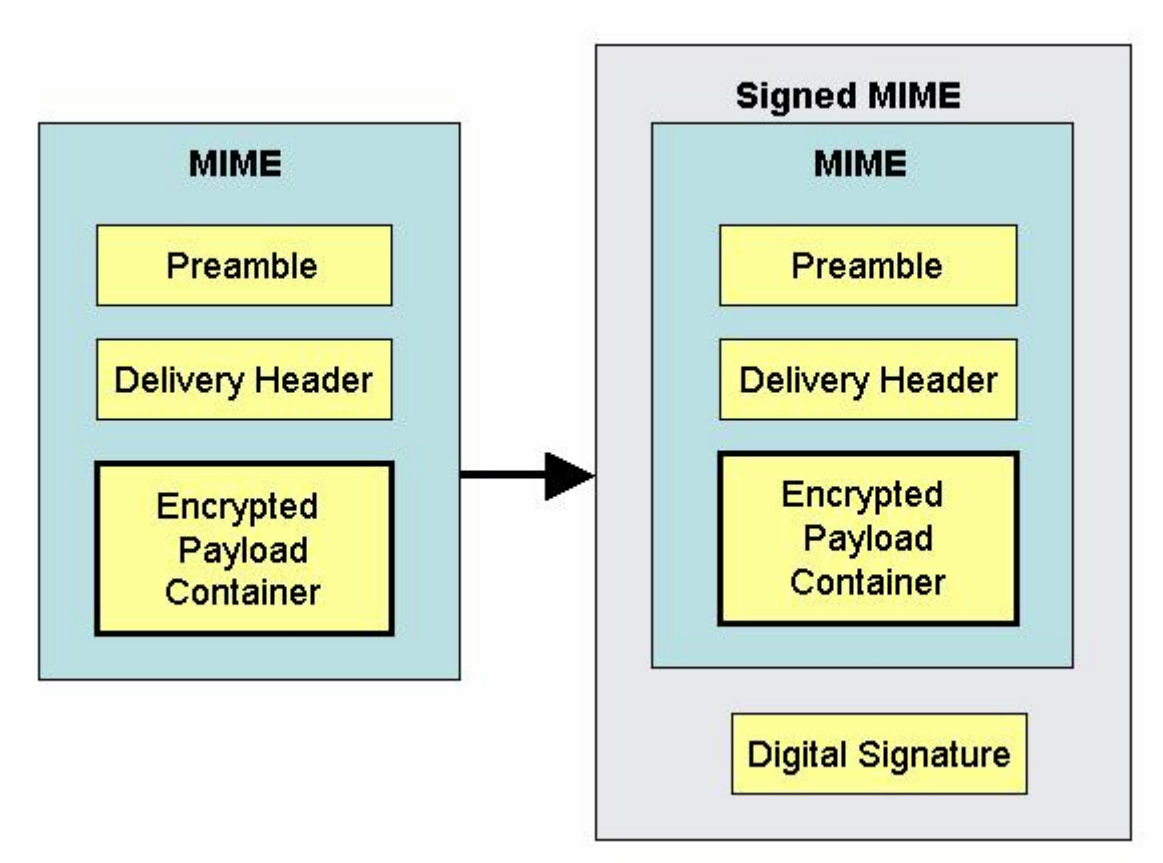

Figure 7: Signed RosettaNet message with encrypted payload packaged in a MIME envelop.

This is a signed payload container encrypted RosettaNet Business message and it is the most complex form of a RN message, see Figure 7. It only leaves the Preamble and the Delivery Header unencrypted.

It is not possible to encrypt the Preamble header since the receiver of the message would not be able to recognize this message as a RosettaNet business message. The Delivery Header is not encrypted because this would make routing impossible. Currently RN only supports peer-to-peer communication but if broadcast or subscription shall be possible, the delivery header must be unencrypted.

When unpacking a RosettaNet business message there are a number of validation and error handling issues to consider. Every part of the message that can be validated against corresponding DTD must be validated. During this phase of message handling it is crucial to detect whether the message sent should be replied in a synchronous or asynchronous matter.

3.4.13 RosettaNet Business Message Transfer

This section specifies protocols used for message exchange and states which are mandatory and which are optional. The implementation of the transferring part of the RosettaNet Business Message exchange is left up to the trading partner with some restrictions. As long as they implement one basic protocol (currently HTTP) it is up to the trading partner to decide which protocol to be used though currently only HTTP and SMTP are regarded as RosettaNet compliant.

3.4.14 Business Signal Specification & Process Control PIP:s

This section of the RNIF specification specifies and identifies PIP:s and business signals used to control the process of the exchange of business messages. Besides the usual PIP:s there is a need to exchange certain system-level messages to aid the processes. These messages are referred to as business signals and perform various system-level administrative tasks.

3.4.15 Flow of RosettaNet Business Messages

This section of the RNIF describes in which fashions messages can be exchanged between trading partners. Much of this is already described in the PIP part of this report so this section will only provide a brief overlook. There is basically three ways of communication used by the RosettaNet standard and that is:

- Asynchronous Single-Action Activity: One message is sent and received by a trading partner. No timeout requirements have to be fulfilled.
- Asynchronous Two-Action Activity: A request is sent to a trading partner and a response is sent from the trading partner back to the initiator. No timeout requirements have to be fulfilled.
- Synchronous Single-Action Activity: One message is sent and received by a trading partner. Timeout requirements have to be fulfilled.
- Synchronous Two-Action Activity: A request is sent to a trading partner and a response is sent from the trading partner back to the initiator. Timeout requirements have to be fulfilled.

One aspect not discussed prior is Timeout. Timeout is specified for different actions in a PIP and it corresponds to the limit of time the trade partner has to produce a reply to a business message [10].

3.5 Security

To unsure secure transfer of business messages between trading partners a variety of different signing and encryption techniques are used. The ones that where encountered during this project are described below.

3.5.1 SLL

SLL or Secure Sockets Layer is a protocol originally developed by Netscape but has over time become an accepted universal standard for authenticated and encrypted messaging over Internet. The SSL protocol runs above the TCP/IP protocol and under high-level protocols like HTTP, LDAP and IMAP. It results in an extra abstraction to the messages sent between two communication nodes see Figure 8.

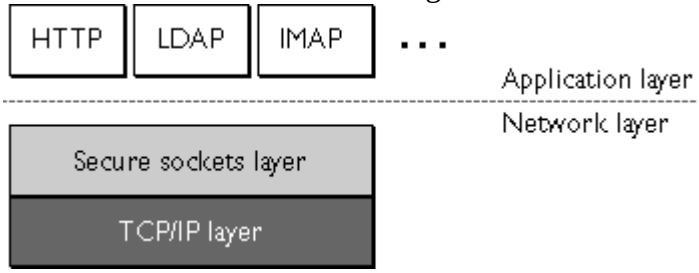

Figure 8: SSL protocol environment [16]

The main functionality of SSL is that it enables a client to authenticate itself to a server and vice versa. It also provides the possibility to establish an encrypted connection between them.

- **SSL Server Authentication:** The SSL client can validate that the servers certificate and public ID are valid and issued by a certificate authority (CA) using public key cryptography.
- **SSL Client Authentication:** The SSL server can validate that the client's certificate and public ID are valid and issued by a certificate authority (CA).
- **Encrypted SSL Connection:** Requires all information to be encrypted by the sender and decrypted by the receiver. This also enables functionality to ensure that the information received is the original message and that it has not been changed or modified during the transfer process.

The SSL protocol is in turn made up of two sub protocols: the SSL record protocol and SSL handshake protocol. The record protocol specifies the formats used to transmit information and the handshake protocol handles the authenticating process [16].

3.5.2 Encryption

There are basically two types of encryption algorithms used and that is symmetric and asymmetric. Asymmetric algorithms may also be referred to as public-key algorithms. When using a symmetric algorithm, the sender and receiver need to have access to a shared key to ensure a secure transfer. When using public key algorithms the sender has access to the receiver's public key and uses this key to encrypt the message. When the message is sent the receiver uses its private key to decipher the message. Since no information has to be exchanged between the sender and receiver this method is regarded as more secure. The drawback compared to a symmetric algorithm is that it is far more computing intensive. Typically a combination of the two algorithms is used in practice. The sender generates a kind of "session" key to be used with a symmetric algorithm, encrypts it using the receiver's public key and transmits it to the receiver. When the receiver has the session key, the sender and receiver can exchange messages between each other using a symmetric algorithm without ever having to expose the shared key to their surroundings [17]. The most commonly used algorithm for key exchange is the RSA key exchange based on the RSA public key algorithm [16].

3.5.3 Signatures

Even though it is possible to use encryption to ensure that no one except the intended receiver can decrypt the message there are still no way to guarantee that the sender and receiver are the ones they give themselves out to be, nor can it be guaranteed that the message has not been tampered with during transmission. To ensure the origin and integrity of the received message they are signed by the sender. Opposed to encryption, the creation of signatures uses a one-way hashing algorithm such as SHA1 [17] to create a message digest. This message digest is then encrypted using the sender's private key and is enclosed in the message. The encrypted message digest is referred to as the signature. When another trading partner receives the message he/she uses the public key from the sender to decrypt the enclosed message digest. This digest is then compared to the message digest calculated from the actual message. If there is a match the integrity and origin of the message is guaranteed.

4 Tools

Many companies present the notion that they can offer a complete integration toolkit in terms of internal and external integration. The challenge of B2B integration software is to provide functionality to connect different applications to different B2B protocols [18]. Since the large number of applications and means of communication used by organizations there is no generic solution to the problem. The five most widespread platforms are the IBM WebSphere Integration Platform, WebMethods Integration Platform, Oracle Integration Solution, Microsoft Biztalk and BEA WebLogic Integration Platform. This report is limited to examine the IBM WebSphere and BEA WebLogic.

4.1 General functionality

There are a few basic qualities that can be requested from RosettaNet software. Besides from the basic properties like being able to perform packing, unpacking, signing, ciphering and validation there are a number of other aspects that has to be considered. For example, what support does the software provide for integration with private processes, how does it handle trading partner information and to what level is it possible to automate the different processes.

There are many alternatives of how to implement a RosettaNet solution and there is no general solution for which one is the mot appropriate one to use. It all depends on the current business landscape, which applications that are used and how they are integrated to each other. Another aspect to consider is which parts of the RosettaNet standard is going to be used and to what extent. Independent of how the business landscape is set up, some mapping between different data structures has to be done. Most enterprise applications has their own set of adapter for RosettaNet and if the business landscape consists of one application, the work of integrating a RosettaNet gateway would merely be one mapping from the internal data structure. However the reality is seldom that simple and most companies use a variety of different business software that implies a more complex integration solution. The integration pattern that fits the organization is then a good hint of what products shall be used and in what manner.

4.2 General Architecture

After looking at a few different integration platforms capable of handling RosettaNet message it is possible to identify which modules have to be present to make up a complete business landscape and the processes that these modules has to contain, see Figure 9. These are listed below:

- **Gateway:** Validation, Trading partner management, Packing and security.
- **Internal Integration Hub:** Mapping, security and backend adapters.
- **Backend systems:** Internal business applications.
- **Internal Communication Backbone:** JMS, MQ or file.

This segmentation is based on Christoph Busslers Execution Semantics of Integration Concepts [18] and the overall structure of the IBM WebSphere concept.

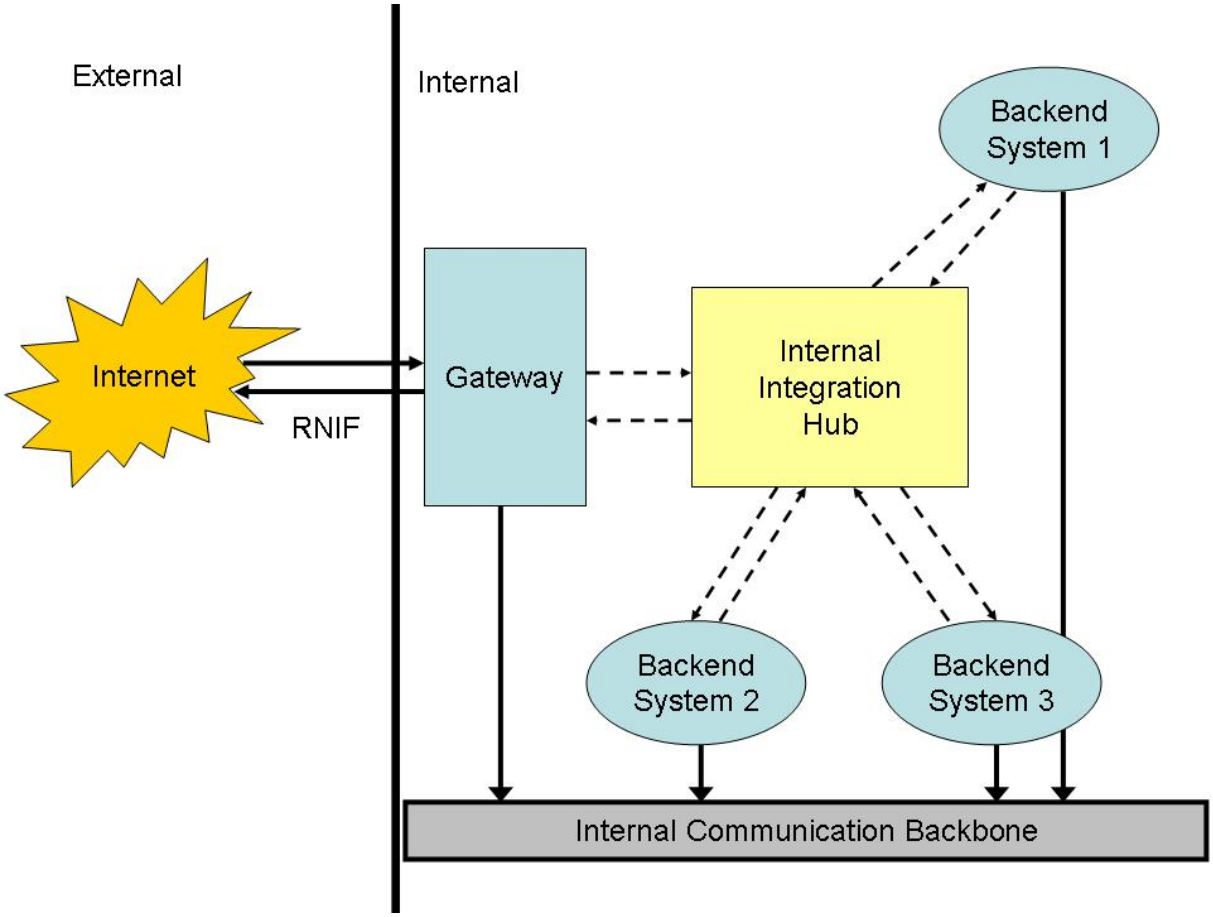

Figure 9: General architecture overview.

4.2.1 Gateway

The gateway is the link between the internal business landscape and the outside world. Every RosettaNet message or any other type of business document that are sent to or received from external sources has to pass through here. For RosettaNet compliance the gateway has to be able to handle validation, packing, unpacking and routing. When a business message arrives at the gateway it has to be unpacked from the MIME envelope, decrypted, verified and validated against the RNIF specification. The gateway has to implement or be in direct contact with a trading partner module that keeps information about the trading partners. The most crucial information is the D-U-N-S number, URL and security information like certificates and keys. The information needed to set up a connection to a trading partner is usually packed together in something that is sometimes referred to as a gateway or channel depending on the product used, but the principle is the same. It is useful to have basic connectivity information bundled with security information since the level of security and the techniques used to provide security could differ between different trading partners.

Every process in the gateway has to be able to generate an exception if it encounters a corrupt message or if some sub process generates an error. When a process is interrupted or terminated the trading partner involved in the business process has to be informed about this. This can be done in two ways, either the exception is sent or the PIP0A1 Notification Of Failure is invoked.

The gateway has to have at least two listeners, one basic URL listener that are used to receive external messages and one internal that can invoke gateway processes when a message is created from the backend system.

4.2.2 Internal Integration Hub

The internal integration hub is a middleware component that handles internal communication. This might not be a stand-alone application if its functionality is implemented in the gateway or the backend system. But for more complex business landscapes this is a separate module. When the gateway has received a RosettaNet message and unpacking is done, the information has to be passed on to the backend system. Since most backend systems cannot handle RosettaNet messages the message has to be mapped to another format. This is where the Internal Integration Hub comes to use. The Internal Integration Hub can hold a number of backend integration adapters and mapping schemas so that the correct backend system can be notified. Sometimes one RosettaNet message holds information that is important to many backend modules, and then the Internal Integration Hub divides and copies the information using different mapping and routing schemas.

This module also has to hold adapters to the backend system to be able to invoke processes that can not be reached otherwise.

4.2.3 Backend System

The backend system in this context basically means the end-point of the message exchange. It can be a customer database (CRM), a business system (SAP) or a separate inventory database. This aspect of a business landscape is out of scope for this report and will only be regarded as an information consumer/producer.

4.2.4 Internal Communication Backbone

This represents the means of internal communication. It can be message queues of any kind (for example JMS and IBM MQ), web services or files in a common storage. The simplest way to realize backbone communication is to use files in a common storage that can be accessed by the different modules. This is a quite primitive way of communication and does not provide any extra functionality that can be achieved with message queues. It is preferred to keep this level of communication independent of the internal message format and that is a large advantage for coming updates and customisations.

4.3 STK

The Self-Test-Kit is a software package supplied by the RosettaNet consortium. It was available for downloading from the RosettaNet homepage www.rosettanet.org. Below is a list of included software:

- RosettaNet Test Engine
- Java JRE 1.3.1: Java Runtime Environment
- OpenSSL: Open source command line application for creation of keys and certificates.
- Apache HTTP Server version 1.3: Standard web server.
- Tomcat Application Server: Java based server for applications.
- JbossMQ JbossMQ: An implementation of the java JMS API

The RNSTK is an application developed by RosettaNet; it is a test system for compliance to the RNIF specification. The purpose of the RNSTK is to provide organizations that want to develop RosettaNet interfaces the ability to validate them against a test application.

This package contains the servlets that is run on the Tomcat application server. It is implemented in Java and is built up by 8 packages.

- Controller: Overhead control of the test process.
- Inbound: Handles incoming messages.
- Outbound: Handles outgoing messages.
- Repository: Storage for PIP specifications and logs.
- Results: Handles the result files and display pages.
- Secure message: Handles security issues.
- Standards: Holds message standards.
- Validators: Handles validation against PIP and RNIF specification.

Below is a high-level diagram of the system where collaboration between functionality in the 8 packages is visualized:

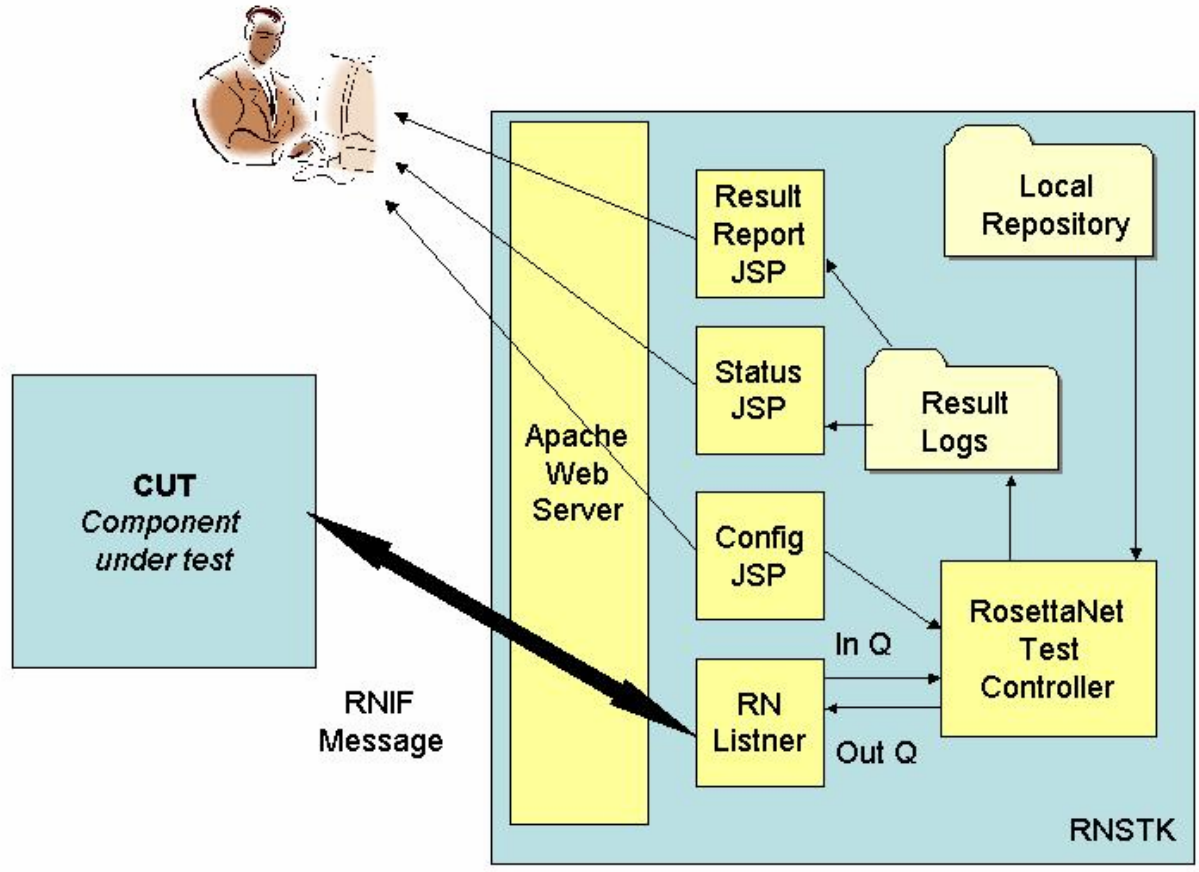

Figure 10: High-level system diagram of RNSTK.

The RosettaNet solution that needs to be tested is in this system referred to as the Component Under Test or CUT. The CUT sends business messages and action signals to the RNSTK (RosettaNet Self-Test-Kit) and these messages and signals are validated by the RNSTK against a given XML-file containing the RNIF BOV. The templates for the PIP DTD:s are stored in a repository on the servers local file system and every message sent to the RNSTK is validated against these templates.

4.3.1 Use Case

A basic use case is that the test operator goes to the current URL holding the RNSTK and logs in. The next step is to make the proper configurations for the test. A dropdown menu for which PIP and which scenario the PIP is to be tested in is presented. There are a number of other different configurations to be made like which kind of encryption to be used. The user can choose from a number of different scenarios for each PIP. The basic case is to have a buyer or a seller that sends a correct business message but the user can also decide to send a corrupt message to the CUT to test its validation and error handling. Figure 11 is a screenshot of the configuration page.

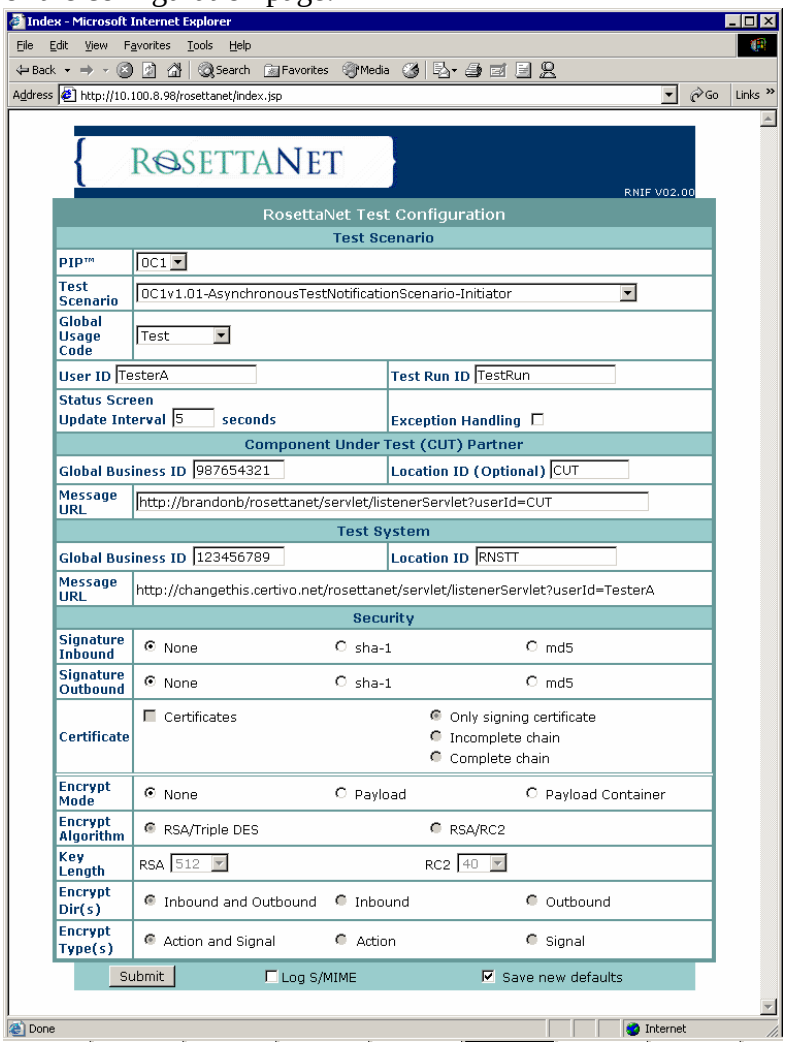

Figure 11: Print-Screen of configuration GUI in RNSTK.

When the configuration is done the test starts. Depending on which PIP that is selected the RNSTK sends a business message to the CUT or is set to a listening state until it receives a message from the CUT. During the test process a status page is displayed where it is possible to monitor the communication between the CUT and the RNSTK. The status page contains information about which business messages that have been sent and if they had correct structure and syntax. If an error is detected an informative error description is presented. When the test is completed the results are saved in a report. The report displays if the scenario was executed correctly or not. If an error was detected, the user gets information about where in the PIP the error was found.

4.3.2 Test Flow:

A more detailed description of the test process is described by Figure 12.

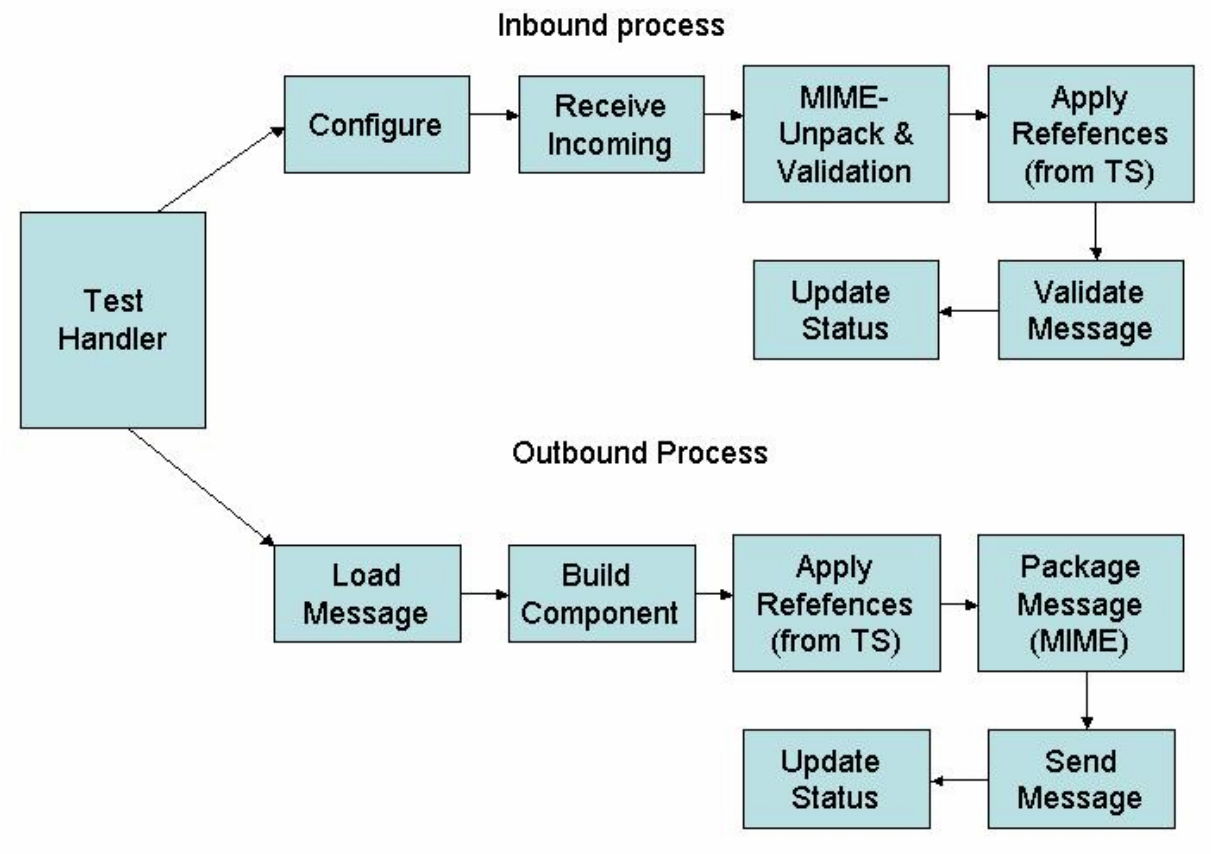

Figure 12: Test flow diagram of RNSTK.

Figure describes how incoming and outgoing messages are handled by the RNSTK. The process is quite complex and requires a lot of logging and validation.

The test begins with the user filling out configuration values and sends them to the server. Then the server starts a listener that waits for a message from the CUT. When the message is received it is processed for unpacking and MIME validation. After this the test system adds references and the message get validated against a RN DTD. When this is done the process is finished and results get saved and the status web page gets updated. The outbound processes are very similar but from another direction.

This is a more detailed representation of the architecture of the RNSTK. There are three storage components representing the file system repository mentioned earlier. Each PIP action has its own validation class in the repository that validates the incoming RN message from the CUT. The validation is conducted using the DTD file for the current PIP action and it is tested in a child-parent manner. All results from the validation are stored in a log that is later on presented to the test operator using the status servlet [19].

4.4 BEA WebLogic

In the latest version of the BEA WebLogic platform there is an example implementation of three different PIP:s. These processes are described and explained in a tutorial that is supposed to work as a guide for users that want to implement own RosettaNet partner gateway solutions. The PIP:s implemented are PIP0A1 Notification of Failure, PIP3A4

Purchase Order Request and PIP3B2 Advanced Shipping Notice. These PIP:s are implemented to get a representation for every different design pattern in the RosettaNet standard. PIP3A4 follows the asynchronous two-action activity and PIP3B2 follows the asynchronous one-action activity. In the current version of WebLogic no synchronous design pattern is implemented. The three implemented PIP:s are supposed to be a template for other PIP:s as well. New PIP:s can be created with minor changes from the already implemented ones. The processes for the different design patterns are quite similar and the main change that has to be done is to construct a new XML schema from the DTD for the PIP and to set the path to that schema.

As discussed earlier, a distinction is made in point to point communication between two business systems when it comes to actions for communication between different trading partners and internal communication. A process that handles internal communication is called private and a process that handles communication between trading partners is called public. The public processes are represented by the PIP implementations and the private processes are example processes that simulate communication with a backend system. The purpose of the private processes in the tutorial is merely to start the public processes and simulate an error. The error causes an exception that in turn results in the sending of Notification of Failure to the involved trading partner.

The basic scenario for the two PIP:s is that a “buyer” sends a request to a “seller” that answer the “buyer” in a predetermined fashion. In the example the “seller side” of the PIP simulates a error in handling of the received data and sends a PIP0A1 Notification of Failure to the “buyer”.

The WebLogic workshop is a tool for implementing processes and the graphical interface gives a good overview of the different building blocks. The workshop also provides a separate interface for mapping of data between different protocols that makes the integration of different backend systems better presented.
Figure 13 below is an illustration of the PIP3A4 “seller side” as used in the WebLogic platform.

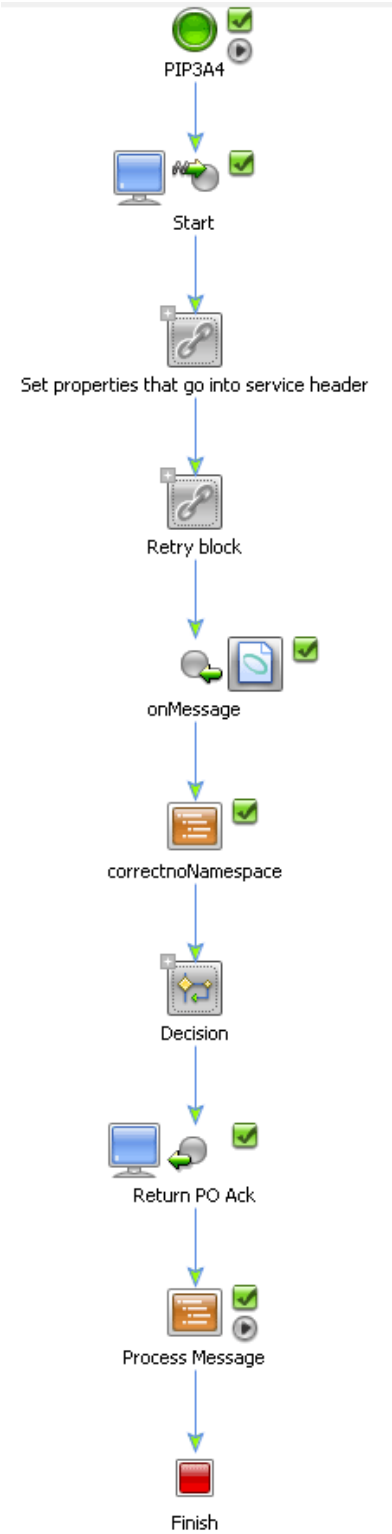

Figure 13: Process diagram of PIP3A4 "seller side" [20].

The trading partner information is handled in another interface called Trading Partner Manager in an integration administration console and these data is used in the PIP implementations [20]. This combined with Data Universal Numbering System D-U-N-S number in the RosettaNet message controls the routing. To change the recipient of a RosettaNet message the D-U-N-S number in the XML file has to be changed manually.

4.5 IBM WebSphere WBI Connect

IBM WebSphere is a product portfolio containing a number of different software products. In this report main focus is on the WBI Connect. WBI Connect is a gateway for business processes that extend over the internal integration landscape. It handles sending and receiving of business messages in a number of different protocols as well as administration of trading partners. To use WBI a few other products are mandatory, such as IBM WebSphere MQ and a database IBM DB2 or Oracle. WBI Connect is built upon the J2EE specification.

4.5.1 System Architecture

The following components make up the WBI Connect product:

- Application servers: Receiver, Document Manager, Community Console
- Queue manager and publish/subscribe broker
- Database
- Common storage

See Figure 14.
Receiver:
As the name implies this component handles receiving of business messages in different formats such as HTTP, files or JMS. The receiver has no direct contact with the database it just copies the received document to a common storage and informs the Document Manager about this.

Document Manager:
This component handles the majority of the workload. The Document Manager handles parsing, validation, signing, encryption, packing, transmission and state management.

Console:
The community Console is basically the GUI for the whole application. It is a web application that handles configuration and monitoring of the system.

Queue manager and publish/subscribe broker:
The queue manager is used to handle communication between the three above-mentioned components.

Database:
The information stored in the database can be divided in to two different categories, profile information and state management. The profile information is mostly configurations that do not change much after the system has been initially configured. The state management handles more active data such as logging.

Common storage:
This is a directory that can be accessed by all other components. In the environment discussed in this report this is only a directory on the hard drive but in a larger production system this would probably be an independent file storage [21].

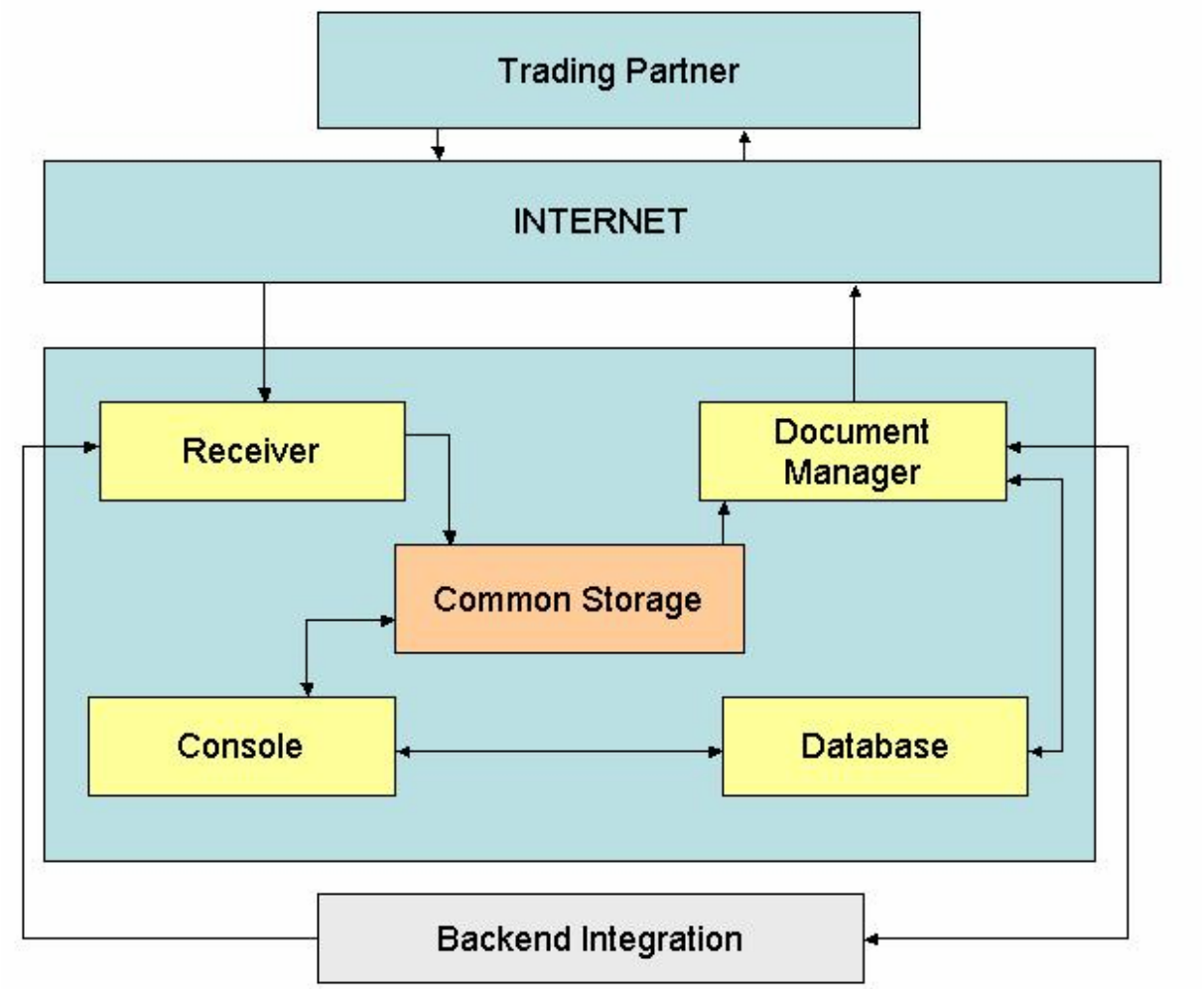

Figure 14: Architectural overview of WBI Connect 4.2.2.

4.5.2 Function/Usage

The administration console is a web browser based human interface. The most central part of the admin console is the account admin. This part of the console handles the profiles for the trading partners. To set up a connection between WebSphere Integration Connect WBIC and a trading partner a few things has to be specified. One or more gateways have do be defined for the trading partner. A gateway can be described as the channel that WBIC uses to perform outbound communication with the trading partner. A trading partner can have many gateways one for each communication channel like HTTP, Java Message Service JMS or FTP. It is also needed to set up a target for the organization holding the WBIC server. The target is a representation of the inbound part of the communication; it can be a JMS queue or a HTTP URL. When these components have been configured the language used for communication has to be specified. Using the Participant Connections part of the console the means of communication is set up. Here it is possible to choose between RosettaNet, EDI or any other B2B communication standard.

To make the WBIC available to the internal business integration landscape the configurations that has do be done is pretty much the same. A new target for the organization has to be set up and a gateway to an application that can handle the information has to be defined. The diagram below shows an example of communication between two organisations both using WBIC, see Figure 15.

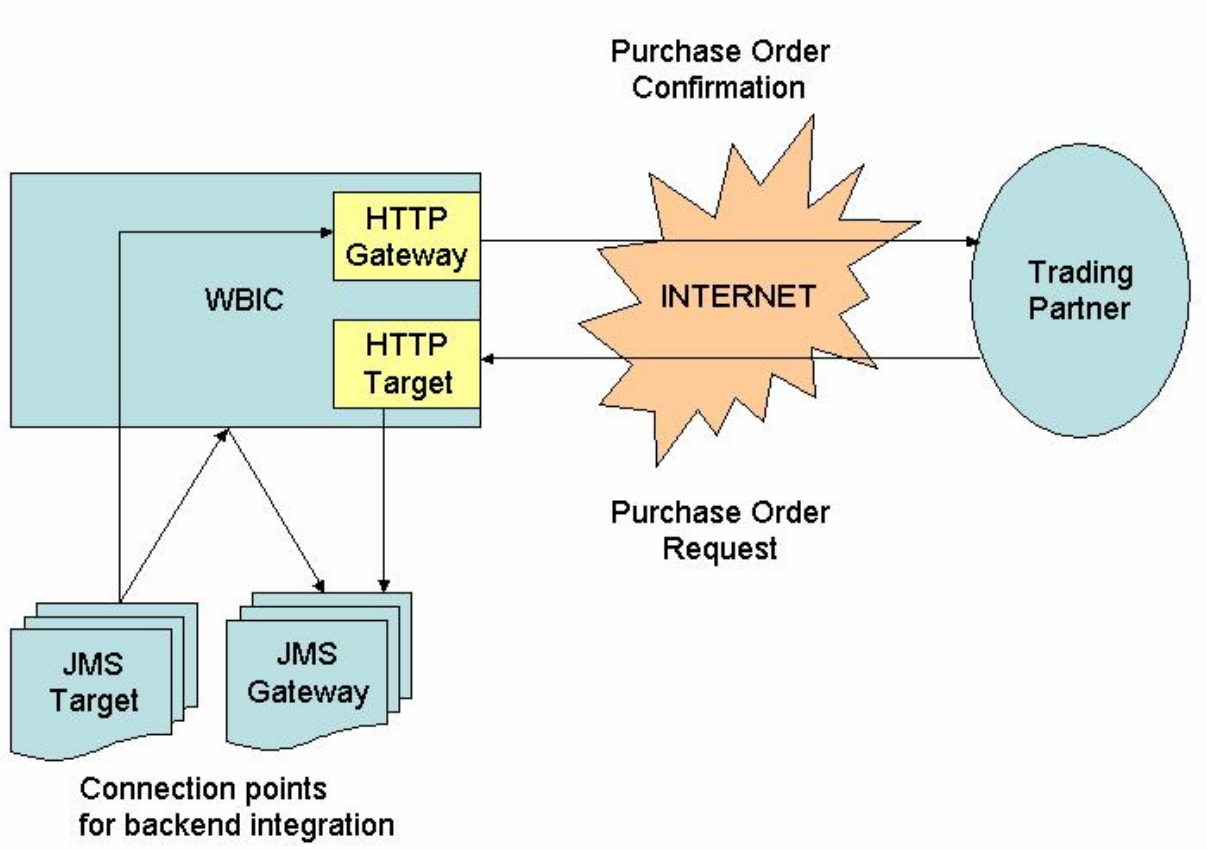

Figure 15: Example of communication between two instances of WBIC.

4.6 IBM WebSphere Interchange Server

The Interchange Server ICS is a server that is used to exchange information between applications using different data structures in a network, internal or external. One of the most central building blocks of the ICS is the collaborations that make up a link between two applications. These collaborations describe the logic of a distributed business processes. ICS and connectors communicate with each other using Business Objects that are the object representation of the serial data required from the current application. The link between the ICS and the applications is called a connector. A connector consists of two parts, a connector controller and a connector adapter. A connector controller communicates directly with the ICS and the connector agent communicates with an application. The connector controller has to be located at the ICS and the connector agent can be located on a different server on an internal network or the Internet. The connectors are part of another IBM product called IBM WebSphere Integration Adapters [22].

Figure 16 is based the Technical Introduction to IBM WebSphere InterChange Server version 4.3.0 and it shows a lot of the possibilities for the ICS.

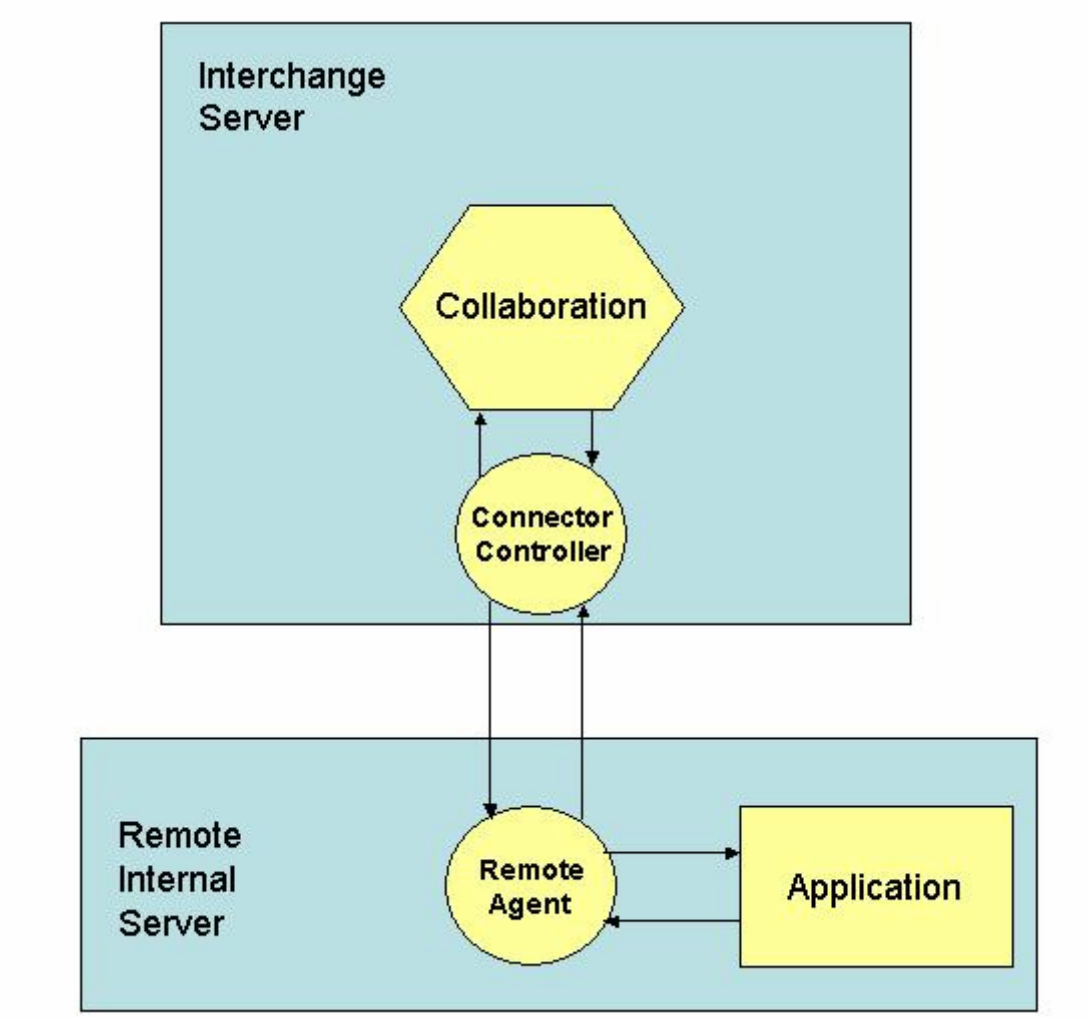

Figure 16: Example of usage of IBM ICS Server and adapters.

ICS uses two types of business objects to represent the business information entity; Application Specific Business Objects ASBO and Generic Business Object GBO. The ASBO:s are used between the Remote Agent and the Connector Controller. The GBO in turn are used inside the Collaboration. The reason to use two or more different representations of the information entity is to provide a more generic solution. This provides the ability to change an application and not having to redo connections from that application to all other applications involved but to just change the mapping and ASBO representation of the native business object. This datamodel is described by Gregor Hohpe et al as the canonical data model [23].

4.7 VM Ware Workstation 5.0

VM Ware Workstation is emulator for operation systems. Once it is installed you can install a variety of operation system on this application. The application manages Virtual Machines on which operating systems are installed and run. It is then possible to take a snapshot of the current virtual machine to use at another computer with WM Ware installed. This is a great advantage if you want to use a system that demands a lot of configuration on many workstations [24].

5 Implementation and Testing

This section describes a modification of the RNSTK and testing of WebSphere and BEA products.

5.1 Modification of RNSTK

The RNSTK is a test client developed by the RosettaNet consortium as described in section 3.2.1. One goal with this project was to modify the RNSTK so that is possible to start a process that are able to receive any kind of PIP and from this PIP construct a kind of "dummy" response to that request so that functionality in the application that initiates the PIP can be verified. This can come to use in early stages of implementing RosettaNet solutions when connection and basic sending abilities have to be verified. It is also possible to extend the module to handle many PIP:s in a row without having to provide basic configurations.

This modification is basically an addition of an extra servlet to the application that handles initial test configurations in a different way.

This extra functionality is added to reduce the number of configurations needed to perform a test of new RosettaNet solution. This makes the test client more similar to other commercial products. And it makes regression-testing possible with only minor configurations.

5.1.1 Functional Requirements

1. The application shall be able to generate a response to any predefined RosettaNet request.
2. The application shall be able to receive any kind of RosettaNet message and provide information about the test procedure to the user.
3. The application shall be able to start the test procedure with an absolute minimum number of configuration parameters.
4. The results from the test procedure shall be saved to a directory-based repository.
5. No modifications can be done to the current classes.
6. As much as possible of the current functions shall be used to implement the process.
7. In a first version the application it will not be bale to handle encrypted and signed messages.

5.1.2 Use Cases

From the above described functional requirements the following user cases can be identified.

1. The user starts the process by providing the following information: Username, CUT name, CUT URL, CUT GBI (D-U-N-S number), if exception handling is used and test run ID.
2. When above data is submitted to the servlet it starts a listener and forwards the users browser to a test status web page where initial configuration info is displayed.
3. When the user receives a request from the CUT the web page is updated and provides information about the current test step.
4. When all test steps are executed the user can choose to display a report from the whole test procedure.

5.1.3 Technical Requirements

The technical requirements are the same as for a basic RNSTK installation. The process is implemented using Java servlet technology and as much as possible of the current classes shall be used to prevent redundancy.

5.2 Design

The new process is called Multi Listener and is placed in the controller module of the application. It is a servlet that is basically a combination of the two existing servlets, Listener Servlet and Configure Servlet. Multi Listener gathers configuration information from the first RosettaNet message request and stores it to a status file opposed to the usual procedure when the user manually provides this data. The servlet then starts the test controller that is the engine of the test process. The test controller reads data from the configuration file and uses the data to start a test process. After this, the procedure works just as it would if the application had been started as in the standard manner. When the initial configurations have been provided by a browser interface a status file is updated to show the initial configuration, in this case a almost empty form. As the servlet requires the first RosettaNet message, this status file is updated and the new data is displayed to the user. Figure 17 describes how the new servlet fit in the standard architecture.

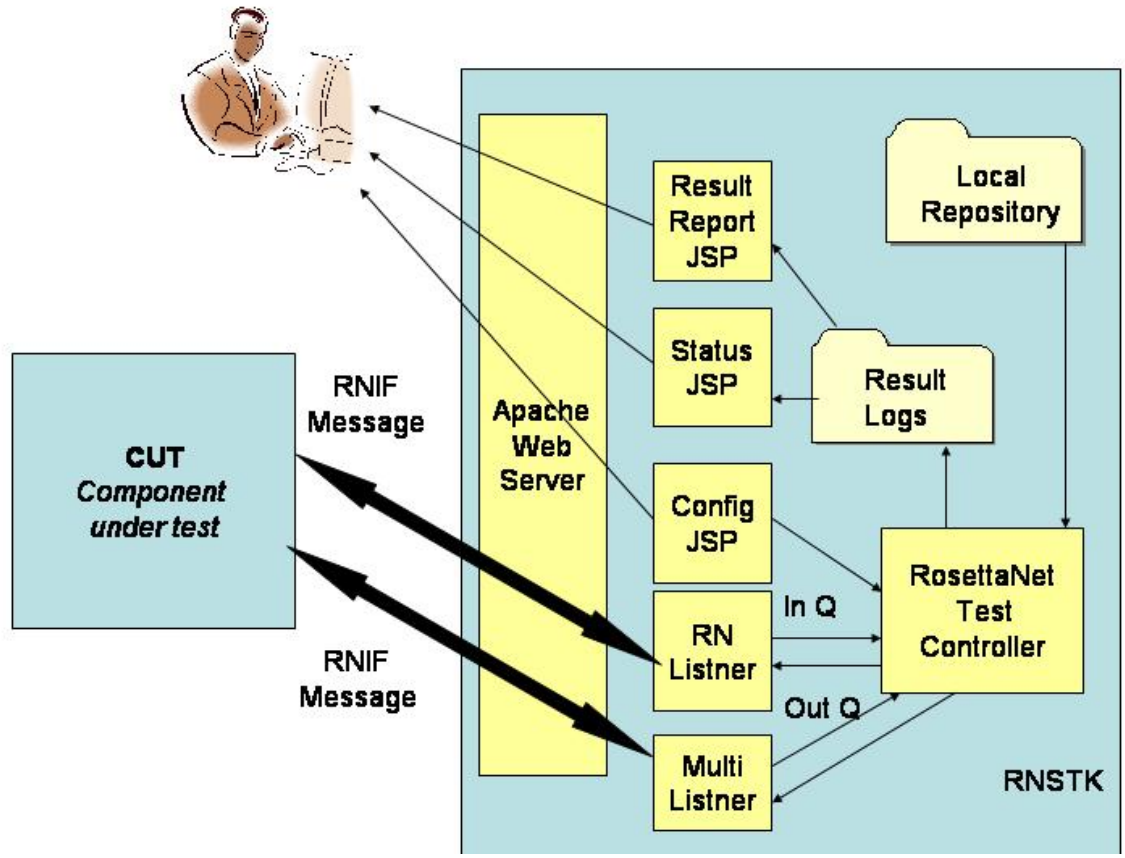

Figure 17: RNSTK with multilistener.

5.2.1.1 Implementation of Functional Requirements

1. The process can initiate the Test Controller to the scenario requested from the CUT. The information is gathered from the first business message that is exchanged. The only prerequisite is that the PIP template files are available in the repository.
2. This requirement is fulfilled providing the PIP template is available in the repository and that the MIME message is not signed or encrypted. The process is able to receive a corrupt RosettaNet message and provide relevant printouts.
3. The absolute minimum amount of information the process needs is the data needed for initialisation that is not provided in a RosettaNet message. These parameters are, CUT URL and user name. Since the CUT GBI is known if the URL is known

this info can be provided manually. The Exception handling parameter is not needed for the process to correctly answer RosettaNet messages but affect the way the results are displayed.

4. Since the process starts the Test Controller in a correct fashion, the resulting functionality in the application works as in the usual cases.
5. This requirement is not strictly fulfilled. One obvious reason for this is that the index JSP page has to be changed to provide the ability to start this new servlet. There is another change made to the Status Servlet to enable to set it in update mode despite that the Test Controller is not running. This is done because otherwise it would not be possible for the user to get the status display to work when started from Multi Listener. The modification has been done in a way that it does not affect the normal use of Status Servlet.
6. This requirement is fulfilled except in one case and that is the functionality in the Listener Servlet. Since the Multi Listener servlet has to have some basic listener properties to receive the first RosettaNet request. The natural way would be to have the Multi Listener servlet to simply forward the request to the Listener Servlet but this cases other problems with the contact with the Status Servlet. For the Status Servlet to function properly it is mandatory that it receives a basic HTTP response object from the users browser so that it can retrieve a print writer to use for status printouts. The servlet also has to use the input stream from the request object that in turns leads to that the request object cannot be forwarded to another servlet since the response and request object already has been used. This means that the listener functionality has to be implemented in the Multi Listener and this would result in redundancy between the Listener Servlet and the Multi Listener.
7. At this point encryption and signing is not available. To make that available some more configurations would be required at initialisation. In the base application this functionality is implemented and these classes could easily be used in the Multi Listener. Since the basic use for this process will be to validate RosettaNet exchange at connection and PIP-syntax the practical use for encryption and signing is not that extensive. Another problem with this is that the certificates provided in the installation have expired and this fact would have made testing complicated or to expensive.

5.2.1.2 Tools used in implementation

Most of the tools used are the same as for the base RNTK application. The only new application is Eclipse 3.0.2 that is used for Java development.

5.2.2 Testing

During development a basic installation of RNSTK is used for testing. Tests have been conducted with both correct and corrupt RosettaNet messages and the process is giving the proper responses to the other client. The process cannot handle encrypted or signed messages and these test cases makes the test engine to throw exceptions in the same way it would if the usual user case had been configured in a corrupt way.

5.3 Commercial Products

5.3.1 BEA WebLogic

BEA WebLogic integration platform offers a large number of possibilities when it comes to building an integration solution with RosettaNet support but will probably result in a lot of

development effort in order to create a complete system. For private business processes WebLogic seems to mainly rely on web services and the interaction on WebLogic processes which in turn leads to that an interface for every other business module has to map data to this format to be able to use the RosettaNet interface. One drawback of WebLogic is that just three example PIP:s are completely implemented and these three PIP:s are supposed to work as a template for other PIP:s. This means that it would require some effort to implement other PIP:s even though the structure and a lot of transformation functionality (encryption/decryption) can be taken from the template PIP:s.

The Trading Partner Manager in WebLogic is a powerful tool for managing data on trading partners and related data. This manager provides the ability to specify which different communication protocols is used by a partner. One great benefit with this is that you can use the same interface even if other standards are used for different trading partners.

WebLogic also has a number of adapters for different backend system such as SAP, PeopleSoft and Oracle. But these are to a large extent just technical functions and a lot of implementation is needed to deploy a fully integrated business landscape.

5.3.2 WebSphere Business Integration Connect / Partner Gateway

The WBI Connect is a very powerful gateway for B2B communication. The IBM WebSphere contains a lot of different software components and a direct comparison with WebLogic Integration platform would not be fare. But a comparison between WebLogic integration platform and WBI Connect can be motivated. WBI Connect and its related software have complete support for a multi tier server environment that in turn contributes to a very scalable system.

WBI Connect is a large program that demands a lot of hardware and software resources. The implementation of the WBI Connect in this project was very time consuming because of software complexity and hardware limitations.

A fully configured and customized WBI Connect server is very powerful. If used in a business landscape built upon other IBM integration components, the level of atomisation and the possibilities of monitoring are quite impressing. This, and the fact that WBI Connect or Partner gateway provides full RosettaNet support, makes it one of the most complete B2B RosettaNet integration systems. The drawback from all of these functions and possibilities is of course that the installation and configuration process of the software is complex and time consuming. As a first time user it is quite hard to keep track of all different versions and fix packs needed during installation. The documentation of WBI Connect is extensive but when one encounter a problem not dealt with in the basic installation guides it is hard to get support form forums and user databases. A full installation of WBI Connect demands a powerful hardware system. The recommendation stated that a minimum of 2 GB RAM, 2 GHz processor and 300 MB hard drive. During the project, tests where conducted on a workstation with 1 GB RAM, 1.6 Pentium M processor and a standard hard drive but since the server where installed on top of a WM Ware workstation 5.0 installation the hardware properties for the virtual operating system where about 700 MB RAM, 700 MHz CPU. Test where also made with a workstation with the following hardware limitations: 460 MB RAM, Pentium II processor and a 40 GB hard drive. The actual server was also in this case installed on top of a WM Ware installation and the properties for the virtual operating system where about 230 MB RAM, 50 % of original CPU capacity. In this case it was not possible to start all three runtime components of WBI Connect. System resources where able to start two of the servers

but when trying to start the third one the system crashed. In terms of hardware needs the most demanding server where the router and the least demanding where the console. Since all three where needed for any relevant usage the testing where dismissed.

6 Evaluation

Below is a description of the pros and cons of the platforms used during this project and in what environment there are thought to function.

6.1.1 RNSTK

The RosettaNet self test kit has been a great aid in this project. I would not consider this a complete commercial product, but then again it is not supposed to be. The installation process was quite complex and it required some third party software. All needed support software was available from the same site as a package. The configuration process where a bit complicated but once the application where up and running it proved to be a nice support tool. The RNSTK is no longer available for downloading from the RosettaNet homepage. This is probably because commercial alternatives now are available. Software development companies now offer tests for RosettaNet solutions that can earn them a RosettaNet compliant badge. The RNSTK where at the time an open-source software product. The reason for this was to support organizations that wanted to test new software and validate its functionality. Now when the RosettaNet standard has gained some more acceptances, the need for a test client is not that widespread and the testing is part of a more commercial environment. During the project RNSTK has proved itself to be a very useful tool, this project has not handled a lot of new implementation of RosettaNet solutions but it has been a small and easy application to validate configurations and installations of other software products at connection and syntax level. The RNSTK is not totally stable and the installation process where quite complex so at this stage it would not be suitable for commercial use. However many of the packages could be used in a new implementation of a RosettaNet solution with minor changes.

For RNSTK to be a complete test client some basic functionality is missing. A part of this project was to implement a modification to the application so it could act as a recipient role without having to specify which PIP to answer a initial configuration. This and the ability to handle consecutive sequence of PIP:s would be useful when testing new RosettaNet solutions. The ability to conduct consecutive sequence testing would not require too much implementing since all basic functionality is already present.

6.1.2 BEA WebLogic Integration Platform

BEA WebLogic Integration platform is suitable for mid-sized organizations with few business processes and a not to complex backend system. The platform has the ability to handle a large number of trading partners with different communication protocols. If the integration solutions in the current business landscape are based on web services, this platform has a lot of benefits. The development workshop is a great tool for process management makes customisations and addition of new processes possible.

The major drawback of the WebLogic RosettaNet implementation is that it is not complete. Even if the three template PIP:s are implemented, some work would be required to provide a RosettaNet solution implementing a few PIP:s. Once the PIP is started it is possible to follow the process from a web page displaying the results. This gives some information about the process but it feels quite inadequate when it comes to validation of RosettaNet structure and

syntax. Much of the sample files seem to have been taken from the RNSTK described earlier in this report.

Running PIP:s between RNSTK and WebLogic is quite carefree and many of the default values are the same. Running WebLogic against WebSphere is a totally different matter and it gives insight in the level of validation in the different applications. Messages that passed WebLogic without complaints where instantly stopped in WebSphere. Some very basic validation steps were not implemented in WebLogic like controlling the function code of the RosettaNet message, if it was a request on a response. Some structural issues were also found like that the phone number where misplaced in the partner description was not noticed by WebLogic. The fact that the messages that are sent from WebLogic is not structural correct is not an issue since these messages are intended to be generated by a backend system and the ones present only are dummy examples. The fact that the system does not validate the messages according to the PIP specification is a much bigger issue since this can result in that corrupt information is forwarded to a backend system. And even though WebLogic can handle the corrupt information the backend system may rely on that the information is correct for use in internal processes.

There are two aspects to take into consideration when evaluating WebLogics RosettaNet support. One is to look at it as a RosettaNet compliant product and in that sense the application leaves quite a lot to whish for. Another way is to look at it as a developing tool with an extensive RosettaNet tutorial, and in that sense it is a very potent tool. BEA is intended it to be an extension of their current development tool and not a complete product.

One great benefit with WebLogic is that it is really simple to install and the tutorial guides and other documentation is of high quality. It is also quite easy to find information and answers on development forums. The only drawback in terms of usage is the encryption and signing functionality that did not work as described in the tutorials.

6.1.3 IBM WebSphere Integration Connect and Partner Gateway

WebSphere Integration Connect and Partner Gateway are basically two versions of the same application. In the following text WBIC is used to describe the application.

As described earlier WBIC is a gateway to the outside world from the internal business landscape. Among the gateway applications evaluated in this project this is by far the most complete. It is constructed to be able to handle a large amount of transactions and has quite a complex architecture. To be able to install WBIC, DB2 ver 8.1 with fixpack 2 or higher and MQ ver 5.3 with service pack CSD08 has to be installed prior to the main installation. Since both MQ and DB2 are separate applications it is possible to install them on separate servers. This means that it is possible to some extent customize the hardware dependencies. In the demo environment used in this project all of the three applications were installed on the same server. In a production environment these applications would probably be residing on different servers. Since WBIC itself is implemented as three servers they can be physically separated from each other as long as a common storage is provided. This provides an advantage for security planning. Only the receiver module has to be exposed to the outside world and the Document Manager and Console can safely operate behind a firewall.

WBIC can handle many types of communication protocols besides RosettaNet, like EDI and user defined XML standards. For each trading partner a profile is created that holds information about which protocols can be used and what means of communication that is

available for example JSM and HTTP. The ability to have both EDI and RosettaNet profiles in the same application can be a great advantage during a transition from EDI to RosettaNet.

WBIC has customized the DTD:s supplied by the RosettaNet to XML schemas that provide a higher level of validation. These can be governed from a function in the Console. Generally it can be said that the level of validation in WBIC is much higher than in the other applications examined during this project. When testing WBIC against the RNSTK and WebLogic it was found that RosettaNet messages that passed the validation from both RNSTK and WebLogic where found corrupt in WBIC. The validation level differed in terms of structure of the message and also some of the values in the template message used by both WebLogic and RNSTK where incorrect.

The main advantage with WBIC is that it provides a complete RosettaNet solution with support for almost all PIP:s. WBIC is also has a lot of connectivity options to other applications and especially IBM applications. WBIC combined with IBM ICS and proper adapters for a backend system can provide a very powerful integration landscape.

The reason that WBIC can present a higher level of validation is according to one of the products executive architects Scott Simmons [25] that they have translated the PIP DTDs into XSDs and since XSDs provide far more options for validation, as cardinality and so on. The future has proven him right since the last PIP versions also come defined as XSDs.

7 Conclusion

This section presents conclusions that are not directly linked to the result of the evaluation but more of a general nature.

7.1 Objective

The overall goal of this project is to find a way to configure a system or present an outline for a new implementation that can be used in deployment of a B2B communication solution that implements and verifies the RosettaNet standard. The main focus has been on configuring commercial products and testing them against each other. Two commercial products and one open source product have been tested. The main goal was reached since the test environment was set up and a number of PIP:s were tested. Some limitations were uncounted like the number of PIP:s the applications supported and the use of encryption. It would be unfair to compare the two platforms directly to each other since they have totally different levels of ambition in terms of offering a complete RosettaNet solution. Both platforms were set up to handle a few PIP:s and could be validated against each others and the test client, RNSTK.

The ability to connect the applications to a backend system was also examined but to not tested. The examination was made using connection adapter simulators. The RNSTK is merely a test client and there would be no point to connect it to a backend system. BEA WebLogic Integration Platform has an adapter family for connectivity with backend systems but the adapters only present an API-like connectivity option and would require a lot of programming. The IBM WebSphere family holds apart from the RosettaNet gateway application a separate integration server called Interchange Server that has an extensive support for connectivity with different backend applications.

7.2 RosettaNet

The RosettaNet standard has had some impact on the B2B trading community and will probably have even more impact over the years to come. As the organization continues to grow and more and more business applications begin to present RosettaNet solutions, the RosettaNet standard gains acceptance all over the world. In a more ideological aspect of the standard it provides a set of rules to enable different participants to engage in commerce in a predetermined fashion. The Forbes columnist Virginia Postrel has a long discussion in her book The Future and its Enemies [26] about how rules affect the evolution of technology and society. She specifies a number of dynamical principles for defining rules (or in this case standards) that has proven as successful. These principles are:

- Allow individuals (including groups of individuals) to act according to their own knowledge.
- Apply on simple universal entities that can be combined in a number of ways.
- Admit understandable, lasting and binding undertakings.
- Recognize criticism, competition and feedback.
- Establish a framework in which people can create balanced, competing systems of more specific rules.

These principles seem to comply with the ideology of the RosettaNet consortium which speaks for a strong and widely used standard.

If the RosettaNet standard can stay intact and keep the right scope apposed to EDI and not like EDI mutate into a number of different sub standards it will probably be the preferred standard for many years. The RosettaNet will probably not knock EDI out at once and many

companies that use EDI today will continue to do so. And since one standard does not exclude the other some parallelism will probably be present during a transition period.

7.3 Future Research

7.3.1 RosettaNet Automated Enablement

One of RosettaNet's latest ambitions is to make the standard more accessible for Small and Mid-sized Enterprises (SME). The project is called RosettaNet Automated Enablement RAE, and is currently being evaluated trough a number of different pilot projects. The problem with a full system-system RosettaNet solution is that it requires a lot of investments in integration software and additional hardware systems to implement a B2B gateway. Many SMEs do not have an IT budget that can handle those kinds of investments. Another aspect is that small companies might not even have a backend system but instead rely on spreadsheet programs like Excel. The main goal for the RAE project is to provide means for SMEs to be able to participate in the Internet based B2B community. This has mutual positive effects both for small and larger companies since the B2B investments become more profitable when more organizations engage in the B2B trading partner community. The plan to realize this is to use widely spread data formats that can easily be managed by smaller companies. One example in use today is a format spreadsheet in PDF format that has the ability to validate input data and transforms it to a regular PIP. This might not sound earth shattering but for the company receiving the PIP it can mean a lot of reduction in manual work. Another great advantage with this is that business documents will get a more unified appearance but still have a multitude of languages and complexity [27].

As more and more larger companies adapt the RosettaNet standard and start to notice the cost savings and reduction in manual workload it will be interesting to tie even smaller trading partners to this standard to get full use of their integration investment. And the SMEs are bound to follow in the footsteps of its bigger trading partners and the need for a cheaper less complicated RosettaNet gateway will be substantial.

7.3.2 Impact on Business Processes

The RosettaNet standard in itself is mostly consisting of XML documents and specifications of how to use these and may at a first glance not seem to be that impressive. It is the possibilities that this standard provides that can affect how trade is conducted between business partners all over the world. It makes it possible to automate processes that until now have been done manually and have them fully integrated with current IT infrastructure. The best way of understanding the power of the RosettaNet standard is to look at it as a language. In the same way that English language in some ways has become an international language the RosettaNet is bound to have a similar role when it comes to B2B communication. As the technology evolves it is now possible to build a solution that automatically can achieve quotations from a number of different suppliers, use scripts to find the best price and delivery possibilities and use this information to create an order. It is also possible to produce periodically generated orders according to forecast information. Much of these processes require a lot of customisation of backend systems and a weigh has to be done of the business benefits are achieved compared to the implementation cost. Still the possibility exists and with time the evolution of ERPs will have to provide functionality to realise these processes.

8 Reference list

1. Gunnar Lilliesköld, Komponenterna skyfflas skilda vägar, Elektronik i Norden ,2002-11-25 http://www.edtnscandinavia.com/tek/showArticle.jhtml?articleID=19500955 (2005-08-02)
2. Conan C. Albrecht et al, Marketplace and technology standards for B2B e-commerce, Brimingham Young University, 2004-01-19.
3. Inovis Inc, Introduction to EDI, http://www.ediuniversity.com/intro/, 2005-07-02.
4. OASIS, http://www.ebxml.org, 2005-07-02.
5. Hussein Badakhchani Senior consultant Orbism a BEA partner, Introduction to RosettaNet, http://dev2dev.bea.com/pub/a/2004/12/RosettaNet.html, 2005-05-25.
6. RosettaNet case studies, Intel and Shinko use RosettaNet standards to build forecast-to-cash procurement process, http://www.rosettanet.org/RosettaNet/Doc/0/FSG4VFMC6RA4B47H0FJB0NNPED/IntelShinkoROICaseStudy.pdf, 2005-07-03.
7. Tom Sheldon and Big Sur Multimedia, EDI (Electronic Data Interchange), http://www.linktionary.com/e/edi.html, 2005-07-05.
8. Refsnes Data, W3 Schools, http://www.w3schools.com, 2005-06-13.
9. RosettaNet, PIP3A1 Request Quote specification V02.01, 2003-05-22.
10. RosettaNet, RosettaNet Implementation Framework: Core Specification Release for Validation 13 July 2001.
11. RosettaNet, RosettaNet Business Dictionary (RNBD) V2[1].1, 2002-10-31.
12. RosettaNet, RosettaNet Technical Dictionary (RNTD) V4.0, 2004-09-15.
13. D&B, D&B D-U-N-S FAQ, https://eupdate.dnb.com/requestoptions.html?cmid=EOE100537, 2005-07-12.
14. UCC and EAN Internatinal now GS1, Global Trade Item Number (GTIN) Implementation Guide, May 2004.
15. UNSPC, UNSPC FAQ, http://www.unspsc.org/FAQs.asp#WhatistheUNSPC, 2005-07-12.
16. Sun, Introduction to SSL, Introduction to SSL, http://docs.sun.com/source/816-6156-10/contents.htm, 2005-08-25.
17. S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade & L. Repka, Internet Engineering Task Force IETE, S/MIME Version 2 Message Specification, March 1998.
18. Christoph Bussler, Modeling and Executing Semantic B2B Integration, Oracle Corporation, 2002.
19. RosettaNet,RosettaNet Ready Self-Test Kit (STK) User's Guide Release Version 2.0.7, 2004-09-27.
20. BEA WebLogic, Tutorial: Building RosettaNet Solutions, http://e-docs.bea.com/wli/docs81/tptutorial/rosettanet.html, 2005-07-18.
21. Geert Van de Putte, Jeffrey Brent, Ronan Dalton, Rich Kinard & Benjamin Thurgood IBM, B2B Solutions using WebSphere BI Connect Version 4.2.2 2.2, March 2005.
22. IBM, Technical Introduction to IBM WebSphere InterChange Server version 4.3.0, 2004-09-30.
23. Enterprise Integration Patterns, Gregor Hohpe & Bobby Woolf et al, Addison-Wesley, 2004.
24. WM Ware Inc, Introduction to WM Ware Workstation 5.0, http://www.vmware.com/products/desktop/ws_features.html, 2005-08-01.

25. Partner Gateway seminar, Scott Simmons executive architect IBM, IBM Forum Kista 2005-09-14.
26. Virginia Postrel, The future and its enemies, The Free Press New Yord, 1999.
27. Chris Benedetto, RosettaNet, Telecom Industries Executive Overview: Opportunities in RosettaNet Standardization, 2004-07-10.
28. RosettaNet, RNSTK Documentation Review Figures 8-31 v3, 2001
29. John Cartwright RosettaNet, Accelerating Connectivity with Small and Medium Enterprises, http://www.rosettanet.org/RosettaNet/Doc/0/OGIR58N3HMMK389PFDBL0IQ307/F13-SME-Business%20Case%20Example%20v11.ppt, 2005-08-05.

9 Appendix

9.1 Appendix A: Dictionary

BD	Bussines Dictionary
BOV	Bussines Operation View
BPSS	Business Process Specification Schema
CA	Certificate Authority
CEFACT	United Nations/ECE agency
CPA	Collaboration Profile Agreement
CPP	Collaboration Protocol Profile
CRM	Customer Relationsship Management
CUT	Component Under Test
DTD	Document Type Definition
D-U-N-S	Data Universal Numbering System
ebMS	Messaging Service
ebRIM	Registry Information Model
ebRS	Registry Service
EDI	Electronic Data Interchange
EDIFACT	Electronic Document Interchange For Administration, Commerce and Transportation
ERP	Enterprise Resource Planning
FSV	Functional Service View
GBI	Global Business Identifier
GCI	Global Class Identifier
GPI	Global Product Identifier
GTIN	Global Trade Item Number
GUI	Grapical User Interface
ICS	Interchange Server
IEC	International Electrical Committee
IFV	Implementation Framework View
J2EE	Java 2 Enterprise Edition
JMS	Java Message Service
MIME	Multi-purpose Internet Mail Extension
MQ	Message Queue
OASIS	Organization for the Advancement of Structured Information Standards
PIP	Partner Interface Process
RAE	RosettaNet Automated Enablement
RBM	RosettaNet Bussines Message
RN	RosettaNet
RNIF	RosettaNet Implementation Framework
RNO	RosettaNet Object
RNSTK	RosettaNet Self Test Kit
ROI	Return Of Investment
RPC	Remote Procedure Call
SME	Small and mid-sized enterprises
SSL	Secure Socket Layer
TD	Technical Dictionay
TP	Trading Partner
UN/SPSC	United Nations/Standard Product and Service Code
URL	Uniform Resource Locator
VAN	Value Added Network

WBIC	WebSphere Business Integration Connect
WBIPG	WebSphere Business Integration Patner Gateway
WSDL	Web Services Definition Language
XLS	Extensible Stylesheet Language
XML	eXtensible Markup Language
XSD	XML schema documents