ROSETTANET
POWERED BY GS1 US

Multiple Messaging Services (MMS) Profile for Web Services (WS)

Document Information	
Name	Multiple Messaging Services (MMS) Profile for Web Services
Version	V11.00.00
Date Updated	July 29, 2008

Legal Disclaimer

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

Copyright

Trademarks

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

Additional Disclaimers

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

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

Document Version History

Version	Date	Notes
V11.00.00	29 July 2008	Published as validated specification

Acknowledgement

This document has been prepared by RosettaNet (www.RosettaNet.org) from requirements gathered during the Foundational Program and in conformance with the methodology. Listed below are the legal entities that contributed to the design and development of this specification.

MMS WS Team

Member	Company	Member	Company
Mir Baqar	Cisco Systems	John Cartwright	Intel
Deepak Bhargava	Cisco Systems	Eric Hamer	Intel
Elham Ghassemzadeh	Cisco Systems	Jeremy Morrissey	Intel
Jim Kao	Cisco Systems	Frederic Herzer	Motorola
Abhijeet Ranadive	Cisco Systems	Thaddeus Marcelli	Motorola
Pranav Shahi	Cisco Systems	Mitri Abou-Rizk	Nokia
Gerald Silverman	Cisco Systems	Jeffrey Hutchins	Oracle
Sundar Subramanian	Cisco Systems	Annabelle Marlow	RosettaNet
John Voss	Cisco Systems	Nikola Stojanovic	RosettaNet
Dale Moberg	Cyclone Commerce	Kevin Liu	SAP
Garry Binder	DHL	Mark Schenecker	SAP
Peter Hawtrey	ePSA	Ron Boutell	Sterling Commerce
Martin Evans	Formfill	Suresh Damodaran	Sterling Commerce
Stewart Witchalls	Formfill	Tom Gindrup	TAG Business Tools
Jacques Durand	Fujitsu	Gary Sheetz	Tyco Electronics
Scott Hinkelman	IBM	Bill Wray	Tyco Electronics
Rania Khalaf	IBM	David Smiley	webMethods
Keeranoor Kumar	IBM	Bala Yanamandra	webMethods

Abstract

This document defines the RosettaNet Multiple Messaging Services profile for Web services. The Profile provides requirements, guidance and best practices about how to use a Web services message handling system to transport RosettaNet Business Messages between business partners.

Table of Contents

1 Overview of Intent

1.1 Motivation for Multiple Messaging Services

RosettaNet implementations today require each business partner to support an implementation of the RosettaNet Implementation Framework (RNIF). RNIF is a B2B message handling system intended for all XML message payloads defined within the RosettaNet standards. Although RNIF is fairly robust and is quite widely adopted within the high-tech industry, RosettaNet is looking for alternatives.

The first reason is that RosettaNet sees its own future as a standards body supporting the upper layers relating to business messages and processes, rather than the lower layers concerned with messaging and other infrastructure. Therefore, as a home-grown standard, RNIF is likely to turn into a maintenance burden for RosettaNet in the long run. Secondly, there is great value in adopting message handling systems that serve multiple vertical markets. The presences of standards like RNIF that cater to specific verticals tend to add interoperability complexity and cost to organizations. Lastly, and perhaps most importantly, there is long-term payoff in making use of horizontal message handling systems that stand on their own right simply due to the fact that these standards will take care of their own evolution as times and technologies change. This is also the way to ensure that any evolution of the horizontal standard will result in the least impact to a vertical industry mission. For example, as pervasive devices become more prevalent, the horizontal standards will evolve to extend the reach to these devices thereby largely eliminating the need for the upper layers to worry about it. Consequently, many RosettaNet business partners view transition from RNIF into other horizontal message handling systems as a strategic business requirement.

Multiple Messaging Services (MMS) is a RosettaNet Foundational Program chartered to address the support of RosettaNet XML business messages and business to business (B2B) collaboration over horizontal message handling systems. In its investigation, RosettaNet concluded that Web services, AS2 and ebMS were the three pre-dominant messaging systems for which specifications need to be derived for using them as a RosettaNet Business Message transport.

1.2 The MMS relationship to MCC

MMS also lays the foundation for the separation of the layers of implementation that is missing in the RNIF specification. Specifically, MMS separates what is now commonly understood to be choreography from message exchange. The subject of the MMS specification is message exchange. Message Exchange Patterns (MEPs) are the atomic units to be considered for how messaging will be implemented and may include information flow between two business partners in one or both directions.

Choreography is the process that ties together the multiple MEPs to implement the full semantics of B2B transactions complete with handling of time constraints and exceptions. MMS specification will be complemented with an accompanying specification, Message Control and Choreography (MCC) whose scope will be addressing the choreography related gaps that emerge when the scope of MMS is set against that of RNIF.

1.3 Goals and Scope of this Profile

The focus of this Profile is the specification of how RosettaNet XML business messages, in the context of RosettaNet business processes, will be transported with certain Quality of Service (QoS) guarantees, using Web services as the messaging infrastructure.

While RNIF will serve as a very useful frame of reference for many issues relating to transportation, QoS and exception handling, the goal will not be to mimic or simulate RNIF. Accordingly, although RNIF has specifications as to how to package business messages, how to provide security and ensure message integrity, how to combine multiple messages and so on, these have been used as guidance to look for ways Web services can achieve the same objectives. There is maximum payoff from the use of a horizontal standard like Web services when and only when we use its features the way they were intended. Web services is an atypical messaging paradigm and has its special ways of handling the many transportation and quality issues, especially relating to addressing, security, reliable message delivery, and so on. Web service does not support all the features that RNIF does, but does provide other features RNIF does not.

In this Profile, the scope is to address a single RosettaNet Business Message exchange. A single message exchange may have the following variations:

- The sending of a RosettaNet Partner Interface Process (PIP) business document and the corresponding receipt of either a fault or a business acknowledgement.
- The RosettaNet Business Message schema could be a Community PIP schema or TPIR-PIP schema.
- Only schema PIPs are allowed. Community PIP schema must exist before the RosettaNet PIP can be used in MMS Web Services.
- The sending of a RosettaNet Business Message and the corresponding receipt of a resulting RosettaNet Business Message
- The sending of a request for a RosettaNet Business Message and the corresponding receipt of the requested RosettaNet Business Message
- The specification is intended to be used for RosettaNet Community or TPIR PIP schemas, and may not support business document schemas or architecture of other standards.

Combining several RosettaNet Business Message exchanges to form a business transaction is out of scope for this Profile, and is in the space of choreography which will be handled by the MCC Foundational Program.

1.4 Prerequisites

This Profile assumes a good understanding of the basic concepts of Web services and the underlying standard specifications.

To get started, Appendix A provides **non-normative** information that is helpful for understanding the context of this Profile. It introduces the basic concepts of Web services and relates the concepts to the architectural approach of this Profile for mapping a RosettaNet PIP to Web services.

1.5 Notational Conventions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119]:

MUST	This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute requirement of the specification.
MUST NOT	This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
SHOULD NOT	This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
MAY	This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).

Normative statements of requirements in this Profile are presented in the following manner:

Rnnnn Formal requirement text here.

The number "nnnn" above is replaced by a unique number in the Profile. To avoid conflicts with requirements defined in other profiles, the qualification MMS-WS should be used together with the Rnnnn number to form a unique requirement identifier, for example MMS-WS R0001.

2 Architecture Overview

2.1 Architecture for Mapping a PIP to Web services

The approach presented in this Profile will take a PIP definition and illustrate how the corresponding XML files can be created to encode the PIP's requirements in Web services, as far as message exchanges are concerned. This section provides an outline of this mapping. It also explains how this Profile is organized to provide detail specification of the related mapping rules.

2.1.1 Accommodating Partners of Different Capabilities

Partners participating in a RosettaNet PIP may not always have advanced infrastructure or constant Internet connection. This profile supports business partners with and without service-hosting capability.

Section 3 describes the different IT scenarios supported by this Profile.

2.1.2 Messages and Message Exchange Patterns

As explained in section 1.3, this profile only address a single RosettaNet Business Message exchange which consists of four patterns.

Section 3 of this profile addresses these message exchange patterns in the context of different IT scenarios and business partner capabilities.

Note that for PIP 0A1, the notification of failure (NOF) is assumed to go through another channel – as the specification says that the participant may no longer be there or running by then. NOF is its own PIP, and is mapped in the same manner as any other. No special provisions are made for it.

This profile includes patterns involving 'pure clients', where this type of node is non-invokable, and can only have limited Web services capability. Within these patterns, specific fields within Web services specification structures may be specified for usage differently than for a 'classic', invokable Web services node. See the section on MEPs and message correlation for details on this topic.

2.1.3 Services Descriptions

Each participant in a PIP, with service hosting capabilities, must be described by a WSDL definition that defines its capabilities. Each of these WSDL definitions must contain one or more portTypes describing the services that the business partner offers for a particular PIP.

Section 4 of this Profile specifies rules for the creation of the corresponding WSDL operation definitions based on the XML Schema of the RosettaNet Business Messages that the operation will exchange and the interaction pattern being used.

2.1.3.1 TPIR-PIPs: Specializing a PIP

RosettaNet Automated Enablement [RAE] Trading Partner Implementation Requirements - Partner Interface Process [TPIR–PIP] Requirement documents are specializations of the community RosettaNet PIPs, usually through restrictions to the message definitions or possibly the QoS constraints. In this approach, we provide the steps required in order to define the proper Web services definitions at the WSDL level needed to enable TPIR-PIPs. This is done step-wise from the published specification files for that particular PIP and consists of minor changes to the target namespace of the WSDL definition and the schema data type of the messages carrying the TPIR-PIP RosettaNet Business Message.

2.1.3.2 Supported Bindings

SOAP over HTTP binding is considered as the default binding that each RosettaNet Web service must support.

Supporting attachments is a vital requirement in B2B business messaging. Today, Web services users use Message Transmission Optimization Mechanism (MTOM) or WS-I AP1.0 as two options to support attachments. However, a clear industry choice has not emerged and this specification would therefore allow the choice of either of them as per a Trading Partner Agreement (TPA) between the two business partners.

Since Web service does not provide a standard way for message compression, this specification does not endorse any particular compression technology. Business partners may choose to use specialized extensions that support compression. Compression is outside the scope of this specification.

2.1.4 Quality of Service

Given the different IT scenarios and different business partner capabilities as described in section 3, the QoS requirements are also different.

Section 5 provides a detail specification of how QoS requirements of a PIP should be addressed using Web services

2.1.5 Business Process

Business processes describe the use of several messages as they relate to each other in order to fulfill a defined business goal.

The work in this Profile is targeted towards enabling the definition of business logic around a set of Web services interactions that handle RosettaNet Business Messages. At this point, we are concerned with being able to reliably and securely exchange these messages as Web services. We do not address business processes, but lay the foundation for its enablement. The space of business logic is expected to be addressed by a separate working group in the MCC Foundational Program.

2.2 Using Web services to exchange RosettaNet Business Messages

2.2.1 Exchanging RosettaNet Business Messages

In order to implement a PIP, a user must download the relevant Community WSDL from RosettaNet, depending on the MEP. If RosettaNet has not yet provided Community WSDLs for a particular PIP, then two business partners can create them from the rules provided in this Profile.

The hosting enabled business partner(s) must create an implementation from the given Community WSDL containing the messages that the business partner wishes to receive. The application logic is developed independently from the QoS requirements as these will be taken care of by the required middleware. Each hosting-enabled business partner then publishes the Web service at a URL of its choosing.

In this Profile, we provide the groundwork for exchanging RosettaNet documents over Web services. In practice, one would combine several related RosettaNet Business Message exchanges that are offered by a single (hosting-enabled) business partner and relating to the same business goal into a single WSDL portType that the business partner offers. The set of those operations and their ordering depends on the business logic surrounding them.

2.2.2 Running a PIP

In order to execute a PIP-based exchange, the business partners must exchange the endpoint(s) of hosted services. This will be done by exchanging a WSDL that contains the WSDL <service> element that contains the URL inside its <port> element.

Each side must also be configured to handle the QoS requirements expressed in the WSDLs. If the middleware does not support WS-Policy, the user can manually configure the system to comply with the policy attachments; however, support for the underlying QoS mechanisms, such as WS-Security, is required in order to enact these policies.

Once all is in place, the parties can start exchanging RosettaNet Business Messages using Web services.

3 Supported IT Scenarios and Message Exchange Patterns

3.1 Accommodating Partners of Different Capabilities

Partners implementing a RosettaNet PIP may not always have advanced infrastructure or persistent Internet connection. In this Profile, we support two kinds of business partners:

- A **pure-client business partner** does not host services and cannot be invoked as a service. It can have varying Web service capabilities supporting at least the minimal set of specifications and standards listed earlier in the background section.
- A **hosting-enabled business partner** has comprehensive Web service capabilities, supporting the full set of the specifications and standards listed earlier in the background section. It can host a Web service, and provide reliable, secure interactions using the relevant Web services specifications. A hosting-enabled business partner cannot invoke a pure-client.

Accordingly, we support PIP interactions in two basic IT scenarios:

- **Service to service**: Interactions between two hosting-enabled business partners
- **Pure client to service**: Interactions between a pure-client business partner and a hosting-enabled business partner.

In the following section, the message exchange patterns associated with each of these IT scenarios is described.

3.2 IT Scenario: Service to Service

In the service to service IT scenario, two hosting-enabled business partners interact with each other. The business requirement is for one business partner to send another a RosettaNet Business Message, and in return it receives a Receipt Acknowledgement or Exception for it.

3.2.1 Service to Service One Way Callback

The Service to Service One Way Callback pattern requires all communication to be WSDL abstract layer one way and responses to be sent in separate connections. This pattern is mapped to a WSDL using two one way services with the first accepting a RosettaNet Business Message and the second accepting a Receipt Acknowledgement.

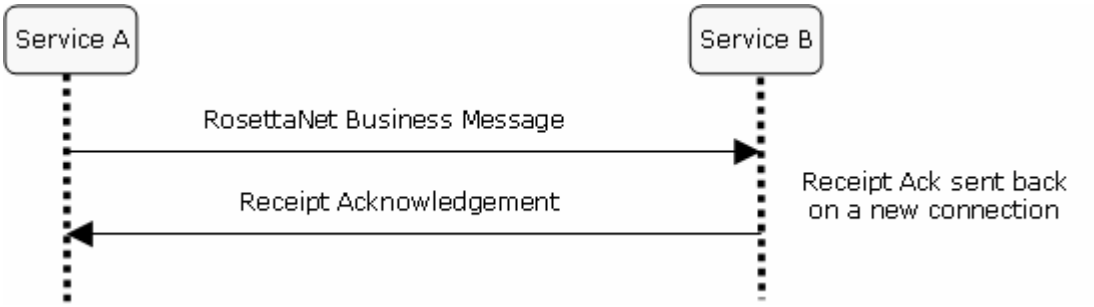


Figure 1: Service to Service One Way Callback Pattern

3.3 IT Scenario: Pure Client to Service

Not all businesses will have the luxury of being able to host a service listener. The barrier may be due to being occasionally connected or that hosting services requires more investment of time and resources and can be a burden to small to medium sized companies. It may also reduce their flexibility. These types of businesses use a pure client to initiate all connections interacting with services to push and pull RosettaNet Business Messages.

As does the service to service patterns, the pure client will still be expected to send and receive single RosettaNet Business Message, but the synchronous nature of the pure client to service interaction can also be used to meet the business need for real-time two action PIPs, such as a Price Check or Purchase Order Request.

Business Requirement	Pure Client To Service Pattern
Pure Client needs to send a service a RosettaNet Business Message, and then receive a RosettaNet Response Business Message or Exception in return.	Pure Client to Service Request Response
Pure Client needs to send a service a RosettaNet Business Message, and then receive a Receipt Acknowledgement or Exception in return.	Pure Client to Service Request Response Push
Pure Client needs to receive a RosettaNet Business Message from a service.	Pure Client to Service Request Response Pull

3.3.1 Pure Client to Service Request Response

The Pure Client to Service Request Response pattern enables a pure client to send a RosettaNet Business Message and receive a RosettaNet Business Message on the same connection. This pattern is mapped to WSDL using one request response service. The pure client will send a RosettaNet Business Message to the service and the service will respond on the same connection with the resulting RosettaNet Response Business Message.

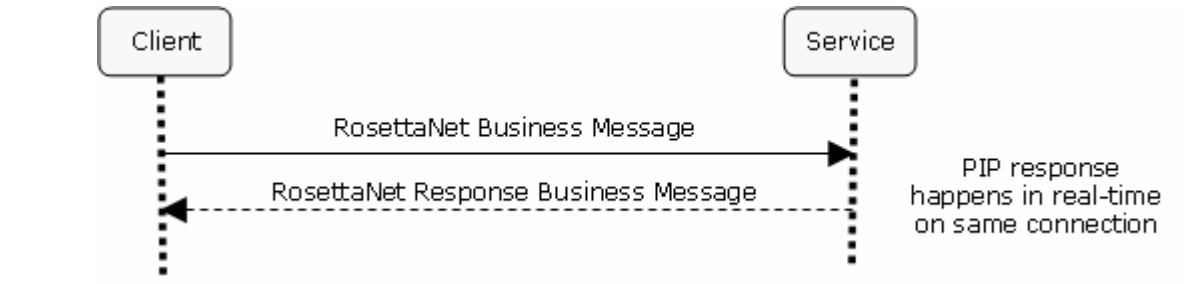


Figure 4: Pure Client to Service Request Response

3.3.2 Pure Client to Service Request Response Push

The Pure Client to Service Request Response Push pattern enables a pure client to send RosettaNet Business Messages to other businesses. This pattern is mapped to a WSDL using one request response service. The pure client will send a RosettaNet Business Message to the service and the service will respond on the same connection with a Receipt Acknowledgement indicating that it received the document.

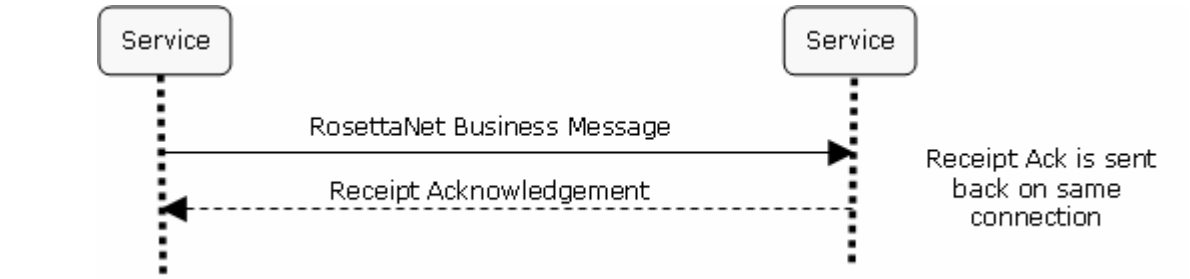


Figure 5: Pure Client to Service Request Response Push

3.3.3 Pure Client to Service Request Response Pull

The Pure Client to Service Request Response Pull pattern enables businesses to send a RosettaNet Business Message to pure clients without requiring the pure clients to host web service listener. This pattern is mapped to a WSDL using one request response service. The pure client will send a soap message request to the service and the service will respond on the same connection with the RosettaNet Business Message requested. To indicate successful receipt of the RosettaNet Business Message, the pure client may initiate a new connection with the service and send a receipt acknowledgment. The client must then look for the HTTPS response code indicating a successful receipt of the Receipt Acknowledgement.

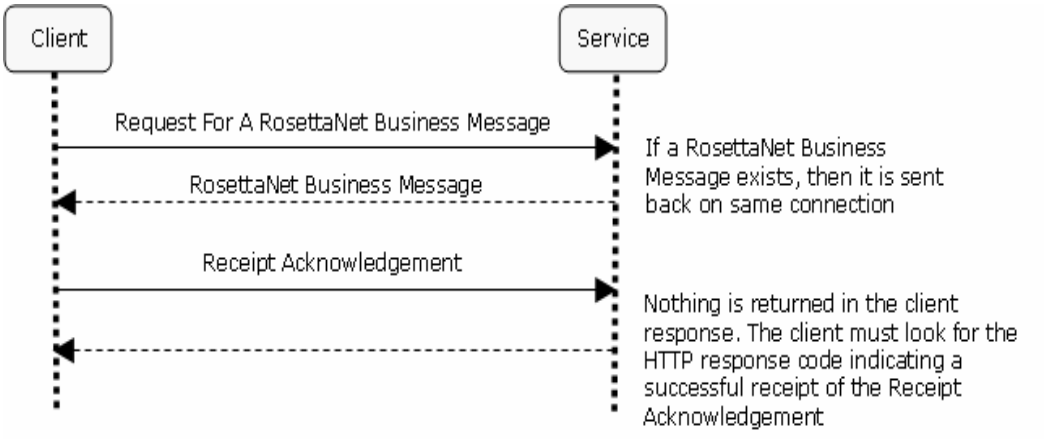


Figure 6: Pure Client to Service Request Response Pull

3.4 Exception Handling

3.4.1 Service to Service One Way Callback

Errors other than SOAP faults relating to the WS-* specifications MUST result in exceptions sent by in separate connection.

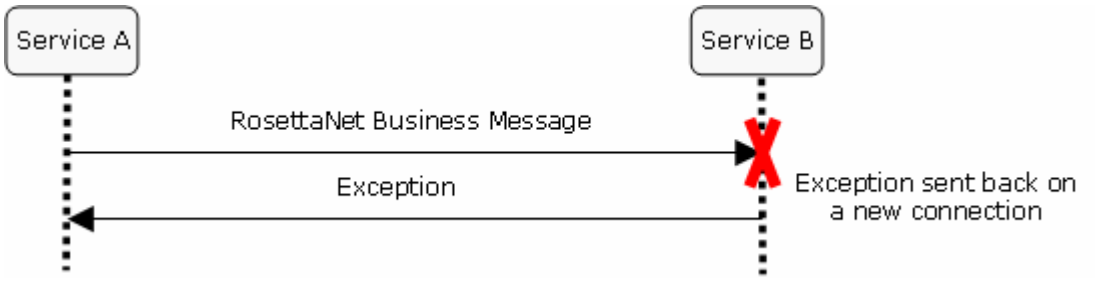


Figure 7: Exception handling in the Service to Service One Way Callback Pattern

3.4.2 Pure Client to Service MEPs

Errors encountered by the service receiving the initiating SOAP request message from the pure client MUST result in exceptions sent back in the same connection.

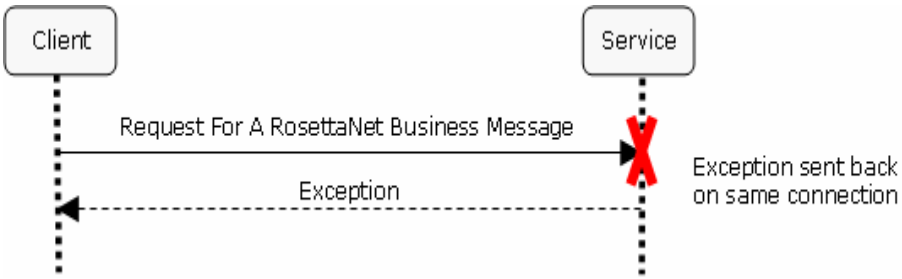


Figure 10: Exception handling in the Pure Client to Service MEPs when the service fails to receive the SOAP request message.

3.5 Message Correlation

This section of the Profile incorporates the following specifications by reference to support message correlation:

- Web Services Addressing 1.0 – Core
  W3C Recommendation 9 May 2006
  http://www.w3.org/TR/ws-addr-core/
- Web Services Addressing 1.0 – SOAP Binding
  W3C Recommendation 9 May 2006
  http://www.w3.org/TR/ws-addr-soap/

Message correlation is the concept of relating one message to another. This is accomplished by identifying each message and using that unique identifier as a correlation token where needed.

3.5.1 Receipt Acknowledgements

R0001 Receipt Acknowledgement (RA SOAP) SOAP messages MUST be correlated to the Soap message containing the related RosettaNet Business Message (PIP SOAP) by setting WS-Addressing Relates-TO in the (RA SOAP) to the WS-Addressing MessageID of the (PIP SOAP)

Rationale
Receipt Acknowledgements are used by the service to track whether the RosettaNet Business Message was successfully received by the recipient and may carry non-repudiation of receipt information. Doing this is not possible without message correlation.

3.5.2 Exception messages / ExceptionOp Operation

R0002 Exception (EM SOAP) SOAP messages sent back to the sender in a separate connection using the ExceptionOp operation MUST be correlated to the SOAP message (SR SOAP) that encountered an error on receipt by setting WS-Addressing Relates-TO in the (EM SOAP) to the WS-Addressing MessageID of the (SR SOAP)

Rationale
Exception messages are used by the service to determine the Soap message that it is related to. Doing this is not possible without message correlation.

3.6 Summary IT Scenario / Business Requirements / Pattern / WSDL

IT Scenario	Message Exchange Pattern	Business Requirement Met	WSDL
Service To Service	One Way Callback	Service needs to send another service a RosettaNet Business Message, and then receive a Receipt Acknowledgement or Exception in return.	WSDL Operations
Pure Client To Service	Request Response	Pure Client needs to send a service a RosettaNet Business Message, and then receive a RosettaNet Response Business Message or Exception in return.	WSDL Operations
Pure Client To Service	Request Response Push	Pure Client needs to send a service a RosettaNet Business Message, and then receive a Receipt Acknowledgement or Exception in return.	WSDL Operations
Pure Client To Service	Request Response Pull	Pure Client needs to receive a RosettaNet Business Message from a service.	WSDL Operations

See Appendix B for more example use cases of the different patterns.

4 WSDL Mapping Rules

This section of the Profile incorporates the following specifications by reference:

- Web Service Description Language (WSDL) 1.1
  W3C Note 15 March 2001
  http://www.w3.org/TR/wsdl

Further, increased interoperability, the following profiles are incorporated by reference:

- WS-I Basic Profile 1.1
  WS-I Final Material 10 April 2005
  http://www.ws-i.org/Profiles/BasicProfile-1.1.html

The use of WSDL 1.1 must follow the WS-I guidance as specified in its related profiles.

From a logical perspective, this profile views WSDL as 3 sections – “What”, “How” and “Where”. This profile is focused primarily on what layer, less focused on the How layer, and does not address the Where layer. The location of services, defined by the Where layer is to be defined by RosettaNet business partners deploying services on to a Web services infrastructure.

In a service to service IT scenario for a single action PIP, where both endpoints are able to receive invocations, two services must be created, one by each business partner. First service provides an endpoint to receive the business document. Second service enables the receiving business partner to either acknowledge the receipt of the business document or indicate that an exception was encountered. Therefore, for a single action PIP, at minimum three WSDL operations must be created. Similarly, a dual action PIP requires six WSDL operations at minimum.

In a pure client to service IT scenario for a single action PIP, one service must be created by the hosting enabled business partner. The hosting enabled partner’s service enables the pure client to either push or pull a RosettaNet Business Message. The service also enables the pure client to acknowledge the receipt of a pulled RosettaNet Business Message. Therefore, for a single action PIP, one WSDL operation must be created for the pure client push. Two WSDL operations must be created for the pure client pull.

4.1 Messages

4.1.1 Importing Message Types

R1001 Types defined in the RosettaNet schemas MUST be imported into the WSDL Type section.

Rationale
The RosettaNet schema defines several complex types, including the type that is used to define the WSDL message, and is provided by RosettaNet workgroups focused on the business content. In order to separate concerns and be consistent in RosettaNet MMS, this profile prohibits authoring of the business content schema directly inside the WSDL Type section.

Example
In case of the purchase order request, the purchase order RosettaNet schema must be imported. It contains the XML Schema types required to exchange message containing the purchase order request.

```
<wsdl:types>
   <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import namespace="…"
schemaLocation="./Interchange/PurchaseOrderRequest.xsd" />
  </xsd:schema>
…
</wsdl:types>
```

4.1.2 Defining WSDL Messages

RosettaNet has defined schema for business messages as well as signal messages. Business message schemas are located in the Interchange folder. Signal schemas are located in the System directory. Following rules apply to all WSDL messages that refer to RosettaNet defined business schemas as well as signal schemas.

R1002 All WSDL messages that refer to RosettaNet business schemas or signal schemas MUST contain a single part.

R1003 If business partners have created TPIR-PIP schema, schemaLocation that refer to corresponding RosettaNet Business Message, MUST refer to TPIR-PIP schema location instead.

Example
<xsd:import namespace="…" schemaLocation="TPIRSchema.xsd"/>

R1004 The single part MUST refer to the root element within the RosettaNet schema.

R1005 The name of the WSDL message MUST be created by adding ‘Msg’ to the local name of the element.

R1006 The part name MUST be created by adding ‘Part’ to the name of the element.

Example - RosettaNet Business Message

- ‘PurchaseOrderRequestMsg’ message contains a single part i.e. PurchaseOrderRequestPart.
- ‘PurchaseOrderRequestPart’ refers to the root element within the RosettaNet schema i.e. poreq:purchaseOrderRequest.
- ‘PurchaseOrderRequestMsg’ is created by adding ‘Msg’ to the root element within the RosettaNet schema.
- PurchaseOrderRequestPart is created by adding ‘Part’ to the root element within the RosettaNet schema.

```
xmlns:poreq="urn:rosettanet:specification:interchange:PurchaseOrderRequest:
xsd:schema:1.0"
.
.

<wsdl:message name="PurchaseOrderRequestMsg">
 <wsdl:part name="PurchaseOrderRequestPart"
element="poreq:PurchaseOrderRequest”/>
</wsdl:message>
```

Example - Receipt Acknowledgement Message

In case of receipt acknowledgment message, the message and part definitions are based on the element ‘receiptack: ReceiptAcknowledgment’.

```
<wsdl:message name="ReceiptAcknowledgmentMsg">
 <wsdl:part name="ReceiptAcknowledgmentPart"
element="receiptack:ReceiptAcknowledgment”/>
</wsdl:message>
```

Example - Exception Message

In case of Exception message, the message and part definitions are based on the element ‘exception:Exception’.

```
<wsdl:message name="ExceptionMsg">
<wsdl:part name="ExceptionPart" element="exception:Exception”/>
</wsdl:message>
```

R1007 An Exception message MUST be used in detail part of SOAP faults for all MEP except Service to Service One Way Callback.

Rationale

In case of all MEPs except Service to Service One Way Callback, exceptions are communicated as SOAP fault detail. In case of Service to Service One Way Callback, ExceptionMsg is used as input to an exception operation (ExceptionOp). Use of ExceptionMsg is SOAP fault, as well as input to ExceptionOp, provides a common exception schema for both the cases.

Example

```
<wsdl:operation name="ProductInformationQueryAndSalesCatalogOp">
…
<wsdl:fault message="tns:ExceptionMsg" />
</wsdl:operation>
```

4.2 Operations

4.2.1 Operation Naming Convention

The operation names are based on the operation parameters. In general RosettaNet PIP Activity gets mapped to operation name, and Action gets mapped to parameters.

R1009 For operations with a single input RosettaNet Business Message (no output), the operation name MUST be constructed by adding ‘Op’ to the root element of the input RosettaNet schema. Following is the convention used:

InputRootElementName**Op**

Example

```
<wsdl:operation name=" PurchaseOrderStatusNotificationOp">
<wsdl:input message="tns:PurchaseOrderStatusNotificationMsg" />
</wsdl:operation>
```

R1010 For operations with an input and output RosettaNet Business Message the operation name MUST be constructed by appending root element of the output RosettaNet schema to root element of the input RosettaNet schema, and then appending ‘Op’ to this name. Following is the convention used:

RequestRootElementName**And**ResponseRootElementName**Op**

Example 1

```
<wsdl:operation name="ProductInformationQueryAndSalesCatalogOp">
<wsdl:input message="tns: ProductInformationQueryMsg" />
<wsdl:output message="tns: SalesCatalogMsg" />
```

```
</wsdl:operation>
```

Example 2
In case of *pure client to service request response pull*, the input message is 'QueryCriterionMsg'. The 'QueryCriterionMsg' may refer to a RosettaNet Business Message specific Query Criterion XSD created based on the template schema (WS_GetMessage_00_01.xsd) defined by RosettaNet, or schema specific to the business partners.

```
<wsdl:operation name="QueryCriterionAndPurchaseOrderRequestOp">
<wsdl:input message="tns:QueryCriterionMsg"/>
<wsdl:output message="tns:PurchaseOrderRequestMsg" />
</wsdl:operation>
```

R1011 Fault operation MUST be named 'ExceptionOp'

Rationale
Input of fault operation is the Exception schema defined by RosettaNet. The operation name is constructed by appending 'Op' to root element of the Exception RosettaNet schema.

Example

```
<wsdl:operation name=" ExceptionOp">
<wsdl:input message="tns: ExceptionMsg" />
</wsdl:operation>
```

R0012 Receipt Acknowledgement operation MUST be named 'ReceiptAcknowledgmentOp'

Rationale
Input of receipt operation is the ReceiptAcknowledgment schema defined by RosettaNet. The operation name is constructed by appending 'Op' to root element of the ReceiptAcknowledgment RosettaNet schema.

Example

```
<wsdl:operation name="ReceiptAcknowledgmentOp">
<wsdl:input message="tns: ReceiptAcknowledgmentMsg" />
</wsdl:operation>
```


4.2.2 Signature of Signal Operations

Signal Operations is used to refer to following two operations: ReceiptAcknowledgmentOp, and ExceptionOp.

R1013 'ReceiptAcknowledgmentOp' MUST have ONLY 'ReceiptAcknowledgmentMsg' as the input

Example

```
<wsdl:operation name="ReceiptAcknowledgmentOp">
<wsdl:input message="tns:ReceiptAcknowledgmentMsg" />
</wsdl:operation>
```

R1014 ExceptionOp MUST have only 'ExceptionMsg' as the input

Example

```
<wsdl:operation name="ExceptionOp">
<wsdl:input message="tns: ExceptionMsg" />
</wsdl:operation>
```

Rationale
In Service to Service One Way Callback MEP errors must be communicated asynchronously using the Exception schema. Therefore, an operation (ExceptionOp) is defined to receive the exceptionMsg.

4.2.3 Operations Required for Mapping Message Exchange Patterns

The WSDL operations and parameters vary with each MEP. In general, each PIP Activity maps to an operation and each PIP Action maps to a WSDL message.

Following table summarizes operations required for each MEP. Rules for this mapping are also detailed below.

As an example, first row of the table should be read as follows. Service to Service One Way Callback MEP involves two services - invoked service (e.g. Buyer service) and initiating service (e,g, Seller service). Invoked service provides a single operation "RequestRootElementNameOp" (e.g. PurchaseOrderStatusNotificationOp) with a single input "PurchaseOrderStatusNotificationMsg". Initiating service provides two operations: ReceiptAcknowledgmentOp, and ExceptionOp. ReceiptAcknowledgmentOp has a single input "ReceiptAcknowledgmentMsg". ExceptionOp operation has a single input "ExceptionMsg".

Message Exchange Pattern	PIPNumber Role	Operation Convention	Example	Message [RootElementNameMsg]
Service to Service One Way Callback	PIP3A7 Buyer [Invoked Service] PIP3A7 Seller [Initiating Service]	RequestRootElementNameOp ReceiptAcknowledgmentOp ExceptionOp	PIP3A7 Buyer PurchaseOrderStatusNotificationOp PIP3A7 Seller ReceiptAcknowledgmentOp ExceptionOp	PurchaseOrderStatusNotificationOp input:PurchaseOrderStatusNotificationMsg ReceiptAcknowledgmentOp input:ReceiptAcknowledgmentMsg ExceptionOp input:ExceptionMsg
Pure client to service request response	PIP2A2 Seller [Invoked Service]	RequestRootElementNameAndResponseRootElementNameOp	PIP2A2 Seller ProductInformationQueryAndSalesCatalogOp	ProductInformationQueryAndSalesCatalogOp input:ProductInformationQuerymsg output:SalesCatalogMsg fault:ExceptionMsg
Pure client to service request response pull	PIP3A4 Buyer [Invoked Service]	QueryCriterionAndRootElementNameOp	PIP3A4 Buyer QueryCriterionAndPurchaseOrderRequestOp ReceiptAcknowledgmentOp (Optional)	QueryCriterionAndPurchaseOrderRequestOp input:QueryCriterionMsg (Schema based) output:PurchaseOrderRequestMsg fault:ExceptionMsg ReceiptAcknowledgmentOp (Optional) input: ReceiptAcknowledgmentMsg
pure client to service request response push	PIP3A4 Seller [Invoked Service] PIP3A4 Buyer [Initiating Service]	RequestRootElementNameAndReceiptAcknowledgmentOp	PIP3A4Seller PurchaseOrderRequestAndReceiptAcknowledgmentOp PIP3A4Buyer PurchaseOrderConfirmationAndReceiptAcknowledgmentOp	PurchaseOrderRequestAndReceiptAcknowledgmentOp input:PurchaseOrderRequestMsg output:ReceiptAcknowledgmentMsg fault:ExceptionMsg PurchaseOrderConfirmationAndReceiptAcknowledgmentOp input: PurchaseOrderConfirmationMsg output: ReceiptAcknowledgmentMsg fault:ExceptionMsg

4.2.3.1 Service to Service One Way Callback

4.2.3.1.1 *Invoked Service*

R1015 For Service to Service One Way Callback scenario, invoked service MUST define at least one business operation.

Example

In following example 'PurchaseOrderStatusNotificationOp' operation is defined.

```
<wsdl:operation name=" PurchaseOrderStatusNotificationOp">
```

R1016 The operation MUST have exactly one input.

Example

```
<wsdl:input message="tns:PurchaseOrderStatusNotificationMsg" />
```

R1017 The input MUST be a WSDL message that refers to RosettaNet business message.

Example

The input refers to RosettaNet 'PurchaseOrderStatusNotification' business message.

```
<wsdl:input message="tns:PurchaseOrderStatusNotificationMsg" />
```

4.2.3.1.2 Initiating Service

R1018 For Service to Service One Way Callback scenario, initiating service MUST define a 'ReceiptAcknowledgmentOp' operation.

Example

```
<wsdl:operation name="ReceiptAcknowledgmentOp">
```

R1019 For Service to Service One Way Callback scenario, initiating service MUST define an 'ExceptionOp' operation.

Example

```
<wsdl:operation name="ExceptionOp">
```

4.2.3.2 Pure client to service request response

4.2.3.2.1 Invoked Service

R1021 For Pure client to service request response MEP, invoked service MUST define at least one business operation.

R1022 The operation MUST have a RosettaNet Business Message as the input, a RosettaNet Business Message as the output, and an ExceptionMsg as the fault.

Example

```
<wsdl:operation name="ProductInformationQueryAndSalesCatalogOp">
<wsdl:input message="tns:ProductInformationQuerymsg" />
<wsdl:output message="tns:SalesCatalogMsg" />
<wsdl:fault message="tns: ExceptionMsg" />
</wsdl:operation>
```

*4.2.3.3 **Pure client to service request response pull***

4.2.3.3.1 Invoked Service

R1023 For Pure client to service request response pull MEP, invoked service MUST define at least one business operation.

R1024 First operation MUST have a QueryCriterionMsg as the input, RosettaNet Business Message as the output, and an ExceptionMsg as the fault.

Example

```
<wsdl:operation name="QueryCriterionAndPurchaseOrderRequestOp">
<wsdl:input = “QueryCriterionMsg”/>
<wsdl:output = “PurchaseOrderRequestMsg”/>
<wsdl:fault = “ExceptionMsg”/>
</wsdl:operation>
```

R1025 The single part of the QueryCriterionMsg MAY refer to the RosettaNet Business Message specific Query Criterion XSD.

The RosettaNet Business Message specific Query Criterion XSD is created by changing the template schema ‘WS_GetMessage’ XML Schema Namespace to the response PIP schema namespace and prefix the PIP root element name found in the namespace with “WS_Get”

Rationale

The template schema ‘WS_GetMessage’ has a single element and provides a standard way to request a message from the invoked service without any search criterion. This template schema has to be modified to be based on the RosettaNet Business Message queried. For example, a Pure Client to Service Request Response Pull WSDL appears as the following for a Community PIP 3A4 Purchase Order Request.

```
<wsdl:message name="QueryCriterionMsg">
<wsdl:part name="QueryCriterionPart" element="c:GetMessage”/>
</wsdl:message>
<wsdl:types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```



```
<xsd:import
namespace="urn:rosettanet:specification:interchange:WS_GetPurchaseOrderR
equest:xsd:schema:01.00" schemaLocation="...."/>
</xsd:schema>
</wsdl:types>
```

R1026 The single part of the QueryCriterionMsg MAY refer to a schema agreed upon within the TPA

Rationale
This provides the business partners the ability to request a message from the invoked service based on a search criterion.

R1026 If business partner specific criterion is used, RosettaNet Business Message specific Query Criterion XSD using the template RosettaNet WS_GetMessage schema MUST be replaced with business partner schema.

R1027 For Pure client to service request response pull MEP, invoked service MAY define a ReceiptAcknowledgmentOp operation.

Rationale
For Pure client to service request response pull MEP, ReceiptAcknowledgment operation is optional since all business interactions may not require non- repudiation of receipt.

Example

```
<wsdl:operation name="ReceiptAcknowledgmentOp">
```

4.2.3.4 Pure client to service request response push

4.2.3.4.1 Invoked Service

R1028 For Pure client to service request response push MEP, invoked service MUST define at least one business operation.

R1029 The operation MUST have a RosettaNet Business Message as the input, ReceiptAcknowledgmentMsg as the output, and an ExceptionMsg as the fault.

Example

```
<wsdl:operation
name="PurchaseOrderRequestAndReceiptAcknowledgmentOp">
<wsdl:input message="tns:PurchaseOrderRequestMsg" />
<wsdl:output message="tns:ReceiptAcknowledgmentMsg" />
<wsdl:fault message="tns:ExceptionMsg" />
</wsdl:operation>
```

4.3 Binding

Expect that both SOAP 1.1 and SOAP 1.2 will co-exist for the near future. This Profile supports either SOAP 1.1 or SOAP 1.2. It is up to the business partner's agreement on which version should be used.

When SOAP 1.1 is in use, this section of the Profile incorporates the following specifications by reference:

- Simple Object Access Protocol (SOAP) 1.1
  W3C Note 08 May 2000
  http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

As constrained by the following WS-I profiles:

- WS-I Simple SOAP Binding Profile 1.0
  WS-I Final Material 24 August 2004
  http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html

- WS-I Attchment Profile 1.1
  WS-I Final Material 10 April 2005
  http://www.ws-i.org/Profiles/BasicProfile-1.1.html

When SOAP 1.2 is used, this section of the Profile incorporates the following specifications by reference:

- SOAP Version 1.2 Part1: Messaging Framework
  W3C Recommendation 24 June 2003
  http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

- SOAP Version 1.2 Part2: Adjuncts
  W3C Recommendation 24 June 2003
  http://www.w3.org/TR/2003/REC-soap12-part2-20030624/

- SOAP Message Transmission Optimization Mechanism
  W3C Recommendation 25 January 2005
  http://www.w3.org/TR/soap12-mtom/

4.3.1 Default binding

R1030 A RosettaNet Web service MUST support at least SOAP over HTTP binding

When no binding is provided for a portType, SOAP over HTTP is the default binding.

4.3.2 MIME binding & Attachments support

Supporting attachments is a vital requirement in B2B business messaging. Today, Web services users use MTOM or WS-I AP1.0 as two options to support attachments. However a clear industry choice has not emerged and this specification would therefore allow the choice of either of them as per a Trading Partner Agreement (TPA) between the two business partners.

R1030 The attachments mechanism SHOULD be agreed upon in the TPA, it can be either MTOM or WS-I AP1.0.

4.3.3 Compression support

Since Web services does not provide a standard way for message compression, this specification does not endorse any particular compression technology. Business partners may choose to use specialized extensions that support compression. Compression is outside the scope of this specification.

4.4 Guidelines for grouping operations

This section provides guidelines on how to group operations into different interfaces. These guidelines are considered optional by this Profile.

Even though a consistent operation grouping mechanism followed by all implementers would be ideal, we recognize that each implementer may have different architectural constraints for interface definitions, and the impact of operation grouping on wire-level interoperability is not as significant as message definitions. Implementers have the freedom to choose their preferred approach for operation grouping and interface definitions as long as they follow WSDL 1.1 specification and WS-I interoperability requirements.

The keyword “MUST” used in this section is only applicable to RosettaNet community WSDLs. Of course, for those companies that choose to follow these guidelines, they may further restrict the profile by making the guidelines mandatory.

The following guidelines apply to all MEPs for a “Single Action Business Interaction” and the “pure client to service request response” MEP for a “Dual Action Business Interaction”. Grouping of operations for “Multiple Action Business Interactions” is not addressed, except the Dual Action scenario noted above.

portType name Mapping is out of scope. Implementer should choose appropriate portType name (e.g. use Role Type as portType name).

4.4.1 Guidelines

G0001 A portType MUST NOT mix operations from different Role Types

Rationale
Business functions in a Single Action Business Interaction are performed by two business partners with two different Role Types (e.g. Sold to, and Sold by). In Web services, the Business Interaction is typically represented as operations of two types - business operations and signal operations. portType is used to group the operations.

Each participant that supports service hosting should provide a portType that groups the operations performed by the participant. This ensures that each participant has a clear understanding about what operations it must provide based on its own role type. Even for a business partner that may support multiple roles with different business partners, it is considered a best practice to provide a separate portType for each role.

Example

Business Interaction: Distribute Order Status

MEP: Service to Service One Way Callback

"Sold to" Role Type Operation: PurchaseOrderStatusNotificationOp

"Sold by" Role Type Operations: ReceiptAcknowledgmentOp, and *ExceptionOp*

ReceiptAcknowledgmentOp, and *ExceptionOp* operations should not be in same port as *PurchaseOrderStatusNotificationOp* because they are associated to different Role Types.

portType A provided by "Sold to" Role Type:

<wsdl:portType name="A">

<wsdl:operation name=" *PurchaseOrderStatusNotificationOp* ">

…..

</wsdl:portType>

portType B provided by "Sold by" Role Type:

<wsdl:portType name="B">

<wsdl:operation name="ReceiptAcknowledgmentOp">

…..

<wsdl:operation name="*ExceptionOp* ">

…..

</wsdl:portType>

G0002 A portType SHOULD contain ALL and ONLY operations of a single MEP, for a Role Type.

Rationale

It is important that each participant has a clear understanding about what operations it must provide based on the MEP. If operations of a MEP are not grouped in the same portType, it will not be clear to a participant which operations should be implemented to support a MEP. If operations of different MEPs are mixed in the same portType, it will not be clear to a participant which subset of the operations should be implemented to support a MEP.

Example for a portType to contain ALL operations of a single MEP, for a Role Type

Business Interaction: Distribute Order Status

MEP: Service to Service One Way Callback

"Sold to" Role Type Operation: PurchaseOrderStatusNotificationOp

"Sold by" Role Type Operations: ReceiptAcknowledgmentOp, and *ExceptionOp*

PurchaseOrderStatusNotificationOp must be in a single portType provided by the "Sold to" party.

ReceiptAcknowledgmentOp, and *ExceptionOp* must be in a single portType provided by the "Sold by" party.

portType A provided by "Sold to" party:

<wsdl:portType name="A">

<wsdl:operation name="*PurchaseOrderStatusNotificationOp* ">

…..

</wsdl:portType>

portType B provided by "Sold by" party:

<wsdl:portType name="B">

<wsdl:operation name="ReceiptAcknowledgmentOp">

…..

<wsdl:operation name="*ExceptionOp* ">

…..

</wsdl:portType>

Example for a portType to ONLY contain operations of a single MEP, for a Role Type.

Business Interaction: Distribute Order Status

MEP1: Service to Service One Way Callback MEP

"Sold to" Role Type Operation: PurchaseOrderStatusNotificationOp

"Sold by" Role Type Operations: ReceiptAcknowledgmentOp, and *ExceptionOp*

MEP2: Pure client to service request response pull

"Sold to" Role Type Operation: None

"Sold by" Role Type Operations: PullPurchaseOrderStatusOp

If the Seller (Sold by) groups *ReceiptAcknowledgmentOp*, *ExceptionOp from MEP1*, and *PullPurchaseOrderStatusOp* from MEP2 in a single `portType`, then a "Sold to" participant is interested in only MEP1 (e.g. Service to Service One Way Callback MEP) will not know which subset of the operations (from the combined `portType`) should be implemented.

Following is **NOT correct** since operations from Service to Service One Way Callback MEP, and Pure client to service request response pull are contained in same `portType`.

portType provided by "Sold by" party:

```
<wsdl:portType name="B">
<wsdl:operation name="ReceiptAcknowledgmentOp">
…..
<wsdl:operation name="ExceptionOp ">
…..
<wsdl:operation name=" PurchaseOrderStatusNotificationOp ">
…..
</wsdl:portType>
```

G0003 Following URN Scheme SHOULD be used for the target namespace (of the community WSDL). Alphanumeric text MAY BE indicative of the business interaction(s).

Structure:urn:rosettanet:specification:interchange:**Alphanumeric**
Text:xml:wsdl:1.0

Rationale
"Alphanumeric text" ensures that the WSDL is not restricted to a single action, and provides flexibility to the group operations related to "*Multiple Action Business Interactions*" in a single WSDL. Note, the above guidelines are intended to be used for community WSDLs created by RosettaNet.

Example

targetNamespace=urn:rosettanet:specification**:**interchange:***3A*:**xml:wsdl:1.0

targetNamespace=urn:rosettanet:specification**:**interchange:***3A4*:**xml:wsdl:1.0

targetNamespace=urn:rosettanet:specification**:**interchange:***PurchaseOrderRequest*:**xml:wsdl:1.0

targetNamespace=urn:rosettanet:specification**:**interchange:***Procurement*:**xml:wsdl:1.0

5 Quality of Services

This section of the profile incorporates the following specifications by reference:

- Web Services Reliable Messaging Protocol
  Submission version to OASIS WS-RX TC, February 2005
  http://specs.xmlsoap.org/ws/2005/02/rm/ws-reliablemessaging.pdf

- Web Services Reliable Messaging Policy Assertion
  Submission version to OASIS WS-RX TC, February 2005
  ftp://www6.software.ibm.com/software/developer/library/ws-rmpolicy200502.pdf

- Web Services Security: SOAP Message Security 1.1
  OASIS Standard Specification, 1 February 2006
  http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

- Web Services Security Username Token Profile 1.1
  OASIS Standard Specification, 1 February 2006
  http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf

- Web Services Security X.509 Token Profile 1.1
  OASIS Standard Specification, 1 February 2006
  http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf

- Web Services Security SOAP with Attachment (SwA) Profile 1.1
  OASIS Standard Specification, 1 February 2006
  http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf

- Web Services Addressing 1.0 – Core
  W3C Recommendation 9 May 2006

http://www.w3.org/TR/ws-addr-core/

- Web Services Addressing 1.0 – SOAP Binding
  W3C Recommendation 9 May 2006
  http://www.w3.org/TR/ws-addr-soap/

- Web Services Security Policy Language
  Submission Version to OASIS WS-SX TC, version 1.1 July 2005
  http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf

Further, increased interoperability, the following profiles are incorporated by reference:

- WS-I Basic Profile 1.1
  WS-I Final Material 10 April 2005
  http://www.ws-i.org/Profiles/BasicProfile-1.1.html

- WS-I Basic Security Profile 1.0
  WS-I Working Group Draft 29 March 2006
  http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

5.1 QoS Design Points

The Web services infrastructure is payload-agnostic. However, any given set of (business) content, in this case RosettaNet PIPs, may present difficulties/mismatches when considering the content design with any given infrastructure design, such as the web services infrastructure. The design of the content may contain features which may be leveraged when using a given infrastructure.

This section is organized consistently with the message exchange patterns defined within this profile. The primary intention of this section focuses upon the service quality considerations primarily with wire formats and configurations identifying what Web services QoS aspects are relevant and applicable in context to the defined patterns.

This profile's scope is constrained to primarily focus on the foundational information message exchanges of RosettaNet Business Messages over Web services, with the intention to be built upon or extended into the space of orchestration of services and formal process flow. The QoS section refrains from specifying aspects outside the scope of this profile. However, in order to provide an increased precision in detail concerning the QoS aspects, this section expresses examples and diagrams needed to establish a comprehensive technical understanding of the abilities and ramifications of the intended information mappings while leveraging the relevant QoS aspects in context of the sometimes substantially varying patterns identified.

5.2 Common QoS

This section addresses Web services QoS aspects which are common across all patterns.

5.2.1 Common Security Policy Usage

WS-SecurityPolicy specification defines a set of assertions relevant to security features in SOAP Message Security, WS-Trust, and WS-SecureConversation. WS-SecurityPolicy is, by definition, a "building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models".

In practice, this means the possible combinations of the features of the security policy language is significant. This profile, while recognizing the emerging state of the WS-SecurityPolicy specification, provides recommendations, considerations, and constraints in specific areas concerning WS-SecurityPolicy.

This profile requires the existence of a WS-SecurityPolicy assertion to specify security requirements for RosettaNet services and the expected wire-level representation at runtime.

R2001 WS-SecurityPolicy document MUST be supported in alignment with the restrictions defined in this profile for defining security QoS for RosettaNet services.

R2002 Attachment of WS-SecurityPolicy assertions to WSDL MAY be supported.

R2003 Attachment of WS-SecurityPolicy assertions to WSDL MUST utilize WS-PolicyAttachment.

R2004 The expected wire representation of the WS-SecurityPolicy assertions MUST be supported.

5.2.1.1 Use of Bindings

WS-SecurityPolicy section 3 defines 3 "bindings" which represent patterns based on the individual security assertions, as a first order of reduction in assertion combinations. For simplicity in context of the defined use cases, this profile prohibits the use of the sp:SymmetricBinding.

R2005 – Bindings within security policies MUST be either sp:TransportBinding or sp:AsymmetricBinding.

For simplicity and consistency, this profile requires the use of the widely implemented RSA-1_5 encryption algorithm for use in transport bindings. In WS-SecurityPolicy, this algorithm is included in the sp:TripleDesRsa15 suite property.

R2006 - All security policies specifying an algorithm suite assertion MUST contain the sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15 element.

The sp:TransportBinding assertion is used to indicate that the message is protected using the means provided by the transport, such as HTTPS. This profile mandates the use of HTTPS as transport security, which is optional in WS-SecurityPolicy.

R2007 - Security policies containing an sp:TransportBinding element MUST contain one sp:HttpsToken element.

5.2.1.2 Use of Tokens

WS-SecurityPolicy defines 10 types of tokens for protecting or associating tokens with the message (sp:UserName, sp:IssuedToken, sp:KerberosToken, sp:SpnegoContextToken, sp:SecurityContextToken, sp:SecureConversationToken, sp:SamlToken, sp:RelToken, sp:X509Token, sp:HttpsToken). For simplicity, and elimination of dependencies on WS-SecureConversation and WS-Trust for this version of this profile, this profile restricts the usage of security tokens to types X509 and HTTPs.

X509 certificates leverage public key encryption where the sender generates ciphertext via the public key in the message recipient's X.509 certificate, and the recipient generates plaintext via its corresponding private key. The message sender has assurance that only the recipient will be able to read the message.

R2008 – Security policies which contain security tokens, the tokens MUST be either an sp:X509Token element (Version 3 token as specified in WSS: X509 Certificate Token Profile 1.0) or an sp:HttpsToken element.

Depending on the security token type, tokens have the ability to indicate message inclusion or its use in security processing. This indicator is optional in WS-SecurityPolicy. For simplicity and consistency, usage of message-level security within this profile requires tokens to be included within the messages.

R2009 – sp:X509token elements within security policies MUST include the sp:IncludeToken attribute with a value of "http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Always".

R2010 –sp:Httpstoken elements within security policies MUST include the sp:HttpsToken/@RequireClientCertificate attribute with a value of "true".

5.2.1.3 Use of Integrity Protection

WS-SecurityPolicy section 5.1 specifies QNames or XPath can be used to specify integrity assertions. In order to reduce complexity, this profile restricts the use of integrity protection assertions to being expressed only as QNames. Individual targeted parts of structures for integrity support are not permitted.

R2011 – Integrity assertions MUST be expressed using QNames.

The sp:SignedParts has ability to require all, or specific headers needing integrity protection, and if the SOAP body requires protection. In order to reduce complexity,

this profile requires that all headers targeted for the SOAP ultimate receiver and the SOAP body of the message is to be integrity protected.

Further, WS-SecurityPolicy permits more than one sp:SignParts elements. A cardinality of 1 element is sufficient to enforce this profile's integrity simplification restriction, and avoids any confusion if more than one is present.

R1012 - All asymmetric security policies MUST have exactly one sp:SignedParts element with zero child elements.

The sp:SignedElements provides the ability to specify specific elements requiring integrity protection. In order to reduce complexity, this profile disallows this capability.

R2013 – All security policies MUST have zero sp:SignElements elements.

5.2.1.4 Use of Confidentiality Protection

WS-SecurityPolicy 5.2 specifies QNames or XPath can be used to specify confidentiality assertions. In order to reduce complexity, this profile restricts the use of confidentiality protection assertions to being expressed only as QNames. Individual targeted parts of structures for confidentiality support are not permitted.

R2014 – Confidentiality assertions MUST be expressed using QNames.

The sp:EncryptedParts has ability to require specific headers which need confidentiality protection, and if the SOAP body requires protection. For simplicity and consistency, this profile views all RosettaNet Business Messages as confidential and requires confidentiality support during exchange. Other supporting information, such as Receipt Acknowledgements are not necessarily required to have such protection.

R2015 - All asymmetric security policies MUST have exactly one sp:EncryptedParts element with zero child elements when the policy is associated with the exchange of a RosettaNet Business Message.

The sp:EncryptedElements provides the ability to specify specific elements requiring confidentiality protection. In order to reduce complexity, this profile disallows this capability.

R2016 – All security policies MUST have zero sp:EncryptedElements elements.

The sp:EncryptedParts/sp:Header provides the ability to indicate that a specific header needs confidentiality protection. In order to provide a high level of protection against a cryptographic attack on a message signature, this profile requires that signatures be encrypted. However, this profile specifies the use of sp:EncryptSignature assertion to support this requirement.

This assertion facilitates confidentiality requirements specifically, and exclusively, targeted toward headers containing signatures. See the section on Signature Protection.

5.2.1.5 Authentication

5.2.1.5.1 Transport-level Basic Authentication

It is important to note that concerning security, this profile is written under the assumption of using WS-SecuityPolicy assertions. At the time of writing, WS-SecurityPolicy does not provide assertions for HTTP basic authentication.

This profile makes no additional assumptions about the semantics of the combination of security assertions of WS-SecurityPolicy – meaning this profile does not define semantics for the absence of any assertions, etc, concerning usage of HTTP basic authentication. Therefore, the use of HTTP authentication is viewed as mutually exclusive to the use of security assertions and the specification of such must be done in a TPA.

R2017 – HTTP basic authentication MAY be used for authentication and SHOULD be specified as such within a TPA.

R2018 – If HTTP basic authentication is used for authentication, HTTPS MUST be used for confidentiality support and SHOULD be specified in a TPA.

5.2.1.5.2 Policy-based authentication

R2019 – X.509 certificates MUST be used for authentication within asymmetric bindings.

R2020 – X.509 certificates MUST be used for authentication within transport bindings.

5.2.1.6 Non-Repudiation of Origin & Receipt

While a digital signature generated by a sender provides the ability of verifying the integrity of the message to detect tampering and to provide sender identity, the use of digital signatures can provide evidence of some action on the associated data facilitating non-repudiation of the action. This is sometimes referred to as proof of possession.

Since this profile mandates the use of X.509 certificates to be indicated within all security policies, the certificate can be used to provide possession of the data through the action of signing by the sender, facilitating non-repudiation of origin. This is the mandated form of non-repudiation of origin used by this profile.

R2021 – Signing using X.509 certificates MUST be used for non-repudiation purposes

R2022 - For non repudiation of origin the entire SOAP message must be signed.

R2023 - For non repudiation of receipt, while the entire SOAP message will be signed, the contained receipt ack business document will hold the hash of the correlated RosettaNet Business document (SOAP body) that was received.

5.2.1.7 Use of Protection Order, and Signature Protection

Protection Order refers to the sequence used for applying security enforcement. The highest level of protection is generally thought to come from signing the body, encrypting it, and then encrypting any signatures.

R2024 –Security policies containing an asymmetric binding MUST contain zero sp:EncryptBeforeSigning elements.

R2025 – Security policies containing an asymmetric binding MUST contain one sp:EncryptSignature element.

WS-SecurityPolicy 7.6 provides an assertion to specify signatures over the SOAP body and SOAP headers must always cover the entire body and entire header elements. It is thought that setting the value of this property to 'true' may help mitigate against some possible re-writing attacks, and to reduce complexity this profile requires the use of this assertion. The default value for this property is 'false'.

R2026 – Security policies containing an asymmetric binding MUST contain exactly one sp:OnlySignEntireHeadersAndBody element.

5.2.1.8 Use of Layout

WS-SecurityPolicy 7.7 provides an assertion concerning layout rules for security headers. This profile does not mandate the use of, or constrain this feature.

5.2.1.9 Examples

An Asymmetric Binding Policy with an X509 token as the initiator token and requirement to always include it in messages, an X509 token as the recipient token and requirement to always include it in messages, the use of the TripleDesRSA15 algorithm suite, a requirement to encrypt signatures (sp:encryptSignature/>), a requirement to sign the message parts before encrypting (expressed as the <sp:EncryptBeforeSigning> element not present), and a requirement to only sign entire headers:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/02/securitypolicy"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.xmlsoap.org/ws/2005/02/securitypolicy:\
Proj\WebServices\docs\securitypolicyJuly2005.xsd">
<sp:AsymmetricBinding>
 <wsp:Policy>
  <sp:RecipientToken>
```



```
 <wsp:Policy>
  <sp:X509V3Token sp:IncludeToken=".../IncludeToken/Always"/>
 </wsp:Policy>
</sp:RecipientToken>
<sp:InitiatorToken>
 <wsp:Policy>
  <sp:X509V3Token sp:IncludeToken=".../IncludeToken/Always"/>
 </wsp:Policy>
</sp:InitiatorToken>
<sp:AlgorithmSuite>
 <wsp:Policy>
  <sp:TripleDesRsa15/>
 </wsp:Policy>
</sp:AlgorithmSuite>
<sp:EncryptSignature/>
<sp:OnlySignEntireHeadersAndBody/>
<sp:SignedParts/>
 </wsp:Policy>
</sp:AsymmetricBinding>
</wsp:Policy>
```

A Transport Binding policy indicating the use of HTTPs and a requirement for the client certificate to be included in the message.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/02/securitypolicy"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.xmlsoap.org/ws/2005/02/securitypolicy:\
Proj\WebServices\docs\securitypolicyJuly2005.xsd">
<sp:TransportBinding>
 <wsp:Policy>
  <sp:TransportToken>
   <wsp:Policy>
    <sp:HttpsToken RequireClientCertificate="true"/>
```



```
  </wsp:Policy>
 </sp:TransportToken>
 </wsp:Policy>
</sp:TransportBinding>
</wsp:Policy>
```

5.3 QoS and Pure Client to Service Patterns

This section addresses Web services QoS aspects which are specifically relevant to each pattern involving a Pure Client invoking a Service.

5.3.1 Pure Client to Service Common QoS

This section addresses QoS aspects common to patterns with pure clients to services.

5.3.1.1 Security

The appropriate security approach for patterns involving a pure client vary on several factors which largely depend on the pure client endpoint capabilities. Since the capabilities of pure client endpoints can not be assumed, consistent security quality aspects can not be specified consistently in pure client patterns.

R2030 – A security policy containing an sp:AsymmetricBinding SHOULD be utilized in patterns involving a pure client.

R2031 – A security policy containing an sp:TransportBinding MAY be utilized in patterns involving a pure client.

R2032 – Patterns involving a pure client MAY utilize HTTP basic authentication.

R2033 – Patterns involving a pure client which utilize HTTP basic authentication MUST utilize HTTPS to provide confidentiality protection.

5.3.1.2 Reliability

Due to pure clients being noninvokable, the WS-ReliableMessaging Protocol can not be utilized between a pure client and an invokable service. Exchanges involving noninvokable endpoints in some cases allows WS-Addressing to be utilized outside the expected use between invokable services.

Pure clients may or may not support aspects of reliable messaging. Handling failures in a standard way enhances reliability in the pure client to service patterns.

5.3.1.3 Retries

Note: This discussion about retries applies to all SOAP messages sent by the pure client to the service

R2034 – When encountering connection failures or other transport related errors, the pure client MAY retry sending the Soap request message that previously resulted in an error

Rationale
Retries are common practice for fault tolerance for connection failures or other transport related errors.

R2035 – The SOAP request message that is retried MUST have a WS-Addressing:MessageID equal to the previous SOAP request message that resulted in an error

Rationale
The service in these pure client patterns can detect duplicate SOAP request messages by comparing WS-Addressing:MessageID values and in certain scenarios greatly improve aspects of reliability, such as eliminating message loss and duplicate messages. Although this rule enables such reliability aspects, this profile considers them as implementation details out of scope of this Profile.

Implementation consideration for services capable of detecting duplicate messages: Services capable of detecting duplicate messages by comparing the WS-Addressing:MessageID to previously received messages can respond back to the pure client using original SOAP response sent back from the original client request. This implies that the service will need to store these original responses or have a way to reproduce them when a duplicate is detected. In order to avoid requiring the service to store the original response for an indefinite amount of time, a retention timeframe should be set by the service. When a duplicate message is detected by the service and the original response cannot be found or recreated due to the retention timeframe being exceeded, then the service can respond back with a SOAP fault indicating that a "duplicate WS-Addressing message has been received".

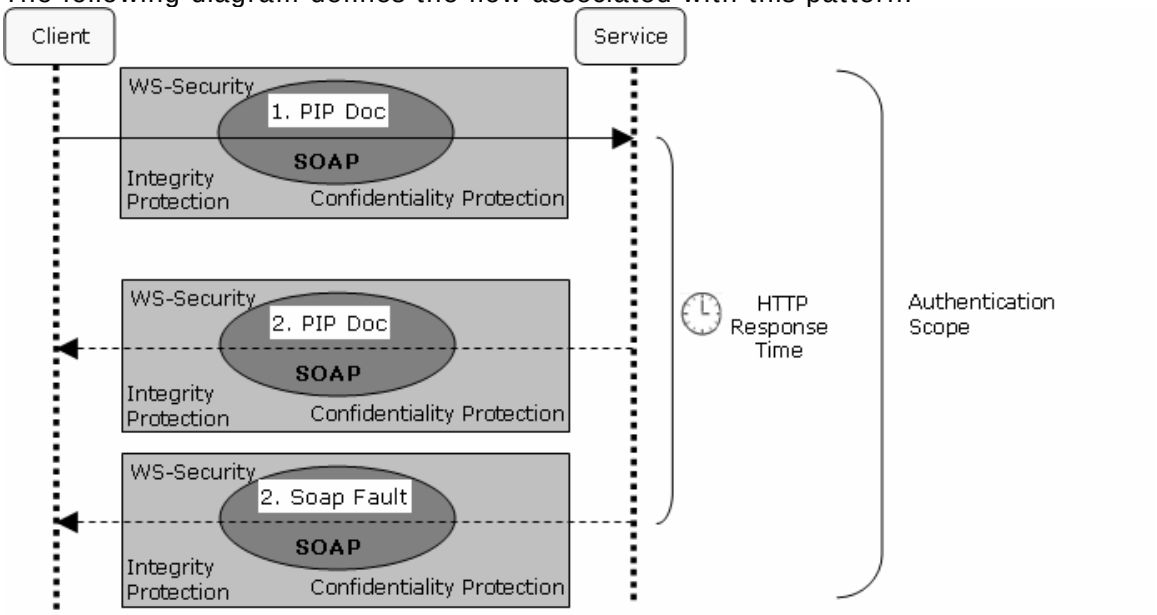
This profile provides specifications on what it passed on the wire when a retry is performed. This profile considers retry thresholds and policies, such as the use of an indicator for maximum number of retries and interval between retries, as implementation detail that would be agreed upon in the TPA if retries are to be used.

5.3.2 Pure Client to Service Request Response Pattern

This environment contains pure client to service interaction. The request contains a RosettaNet Business Message and the response contains a resulting RosettaNet Business Message. The communication is synchronous in the sense that the request is sent and the response is received using the same transport connection.

5.3.2.1 Overall Flow with QoS

The following diagram defines the flow associated with this pattern.



1) Confidentiality protection is required in this exchange. Client invokes Service's operation accepting a RosettaNet Business Message. At completion of sending the RosettaNet Business Message by the client, client starts its HTTP response time clock. The HTTP response time is what is considered a reasonable/acceptable time to leave an HTTP connection open which can vary depending on network latency and message size. At acceptance of the RosettaNet Business Message, Service starts its HTTP response time clock.
2) Within its HTTP response time limit, Service responds using the same connection with either a resulting RosettaNet Business Message or a SOAP fault indicating PIP failure.

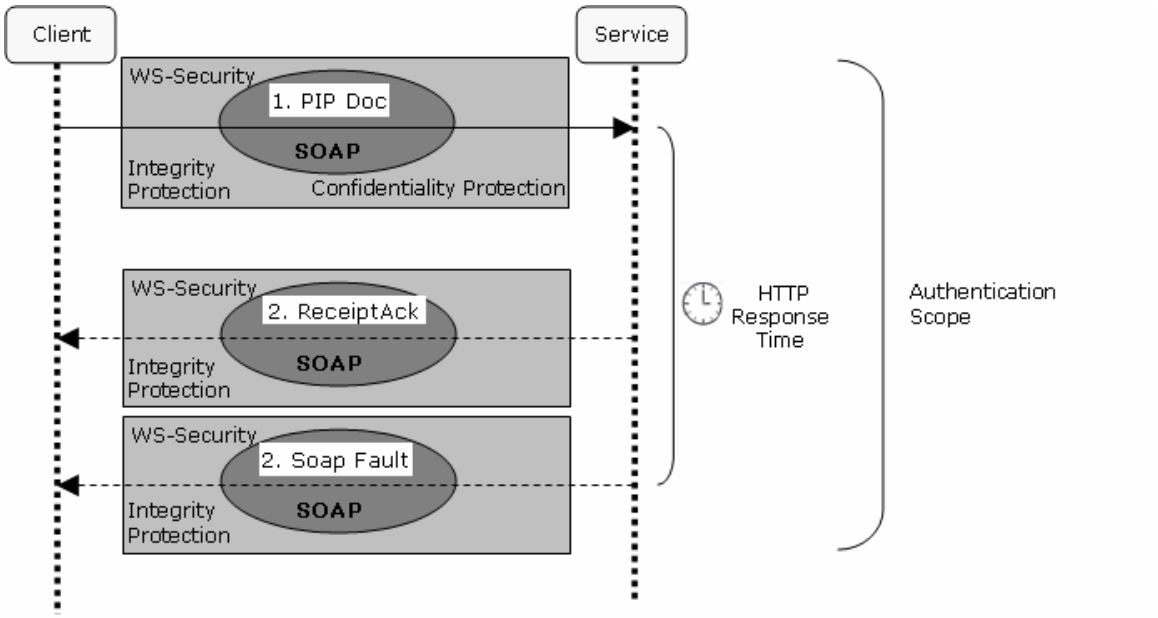
5.3.2.2 Timers

Timers are useful in measuring time and triggering actions when a specific period of time has elapsed. In this pattern, the request and response is on the same connection and the HTTP response time should be no longer than what is a reasonable/acceptable time to leave an HTTP connection open. The duration of the HTTP response time would be agreed upon in the TPA. After such time has elapsed, the client MUST close the connection and treat it as a transport related error.

5.3.3 Pure Client to Service Request Response Push Pattern

5.3.3.1 Overall Flow with QoS

The following diagram defines the flow associated with this pattern.



1) Confidentiality protection is required in this exchange. Client invokes Service's operation accepting a RosettaNet Business Message. At completion of sending the RosettaNet Business Message by the client, client starts its HTTP response time clock. The HTTP response time is what is considered a reasonable/acceptable time to leave an HTTP connection open which can vary depending on network latency and message size at acceptance of the RosettaNet Business Message, Service starts its HTTP response time clock.

2) Within its HTTP response time limit, Service responds using the same connection with either a Receipt Acknowledgement indicating that the RosettaNet Business Message has been received or a SOAP fault indicating PIP failure.

5.3.3.2 Timers

Timers are useful in measuring time and triggering actions when a specific period of time has elapsed. In this pattern, the request and response is on the same connection and the HTTP response time should be no longer than what is a reasonable/acceptable time to leave an HTTP connection open. The duration of the HTTP response time would be agreed upon in the TPA. After such time has elapsed, the client MUST close the connection and treat it as a transport related error.

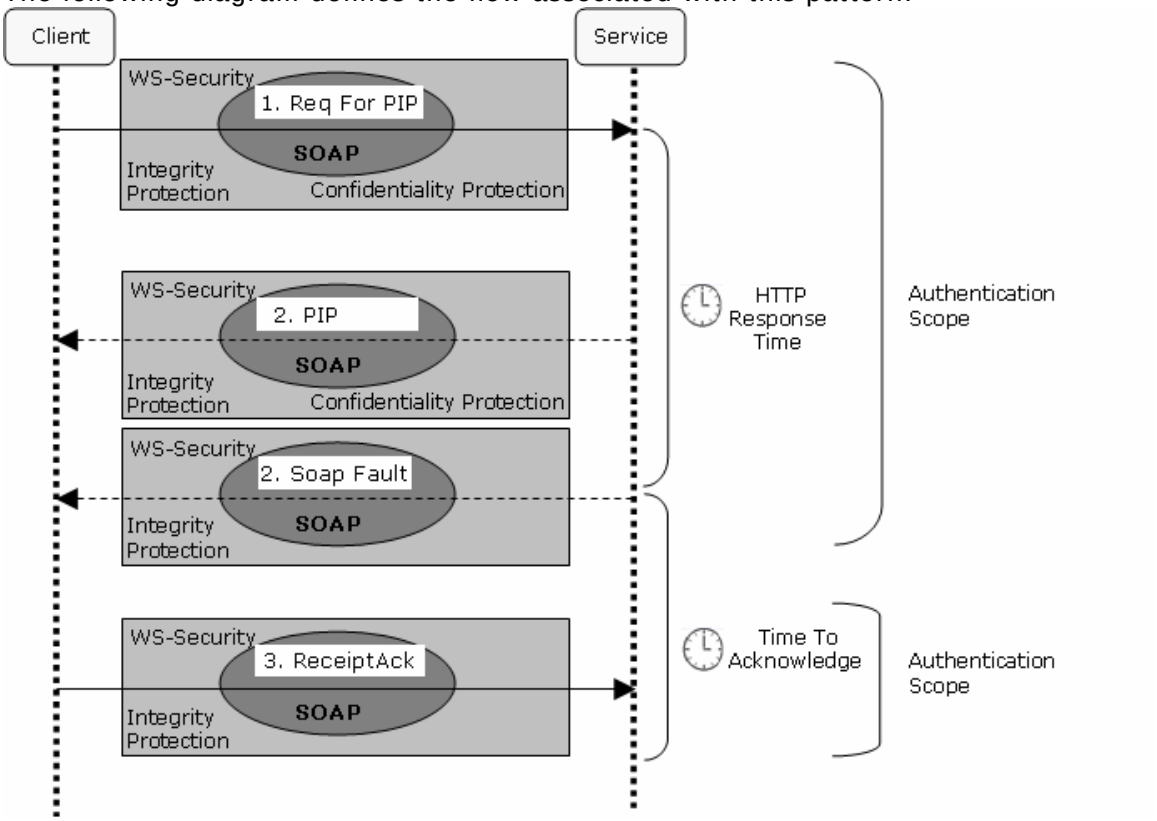
5.3.3.3 Receipt Acknowledgements

Receipt Acknowledgements exchanged indicate that a RosettaNet Business Message was received and it validated against its schema. The Receipt Acknowledgement can provide non-repudiation of receipt when it is digitally signed. They are returned from the recipient to the sender of the message exchange and are used to complete business transactions and support non-repudiation of receipt. The Receipt Acknowledgement in the pure client to service request response push MEP is returned by the recipient (Service) to the sender (Pure Client) on the same connection that the RosettaNet Business Message was sent on.

5.3.4 Pure Client to Service Request Response Pull Pattern

5.3.4.1 Overall Flow with QoS

The following diagram defines the flow associated with this pattern.



1) Confidentiality protection is required in this exchange. Client invokes Service's operation to send a request for a RosettaNet Business Message. At completion of sending the request message by the client, client starts its HTTP response time clock. The HTTP response time is what is considered a reasonable/acceptable time to leave an HTTP connection open which can vary depending on network latency and message size. At acceptance of the request message, Service starts its HTTP response time clock.

2) Within its HTTP response time limit, Service responds using the same connection with either a resulting RosettaNet Business Message, an empty/null message indicating no RosettaNet Business Messages to receive, or a SOAP fault indicating a failure processing the request. If the Service responds back with a RosettaNet Business Message, then the Service will starts its PIP Time To Acknowledge clock at completion of sending the RosettaNet Business Message by the service. At acceptance of the RosettaNet Business Message, client starts its PIP Time To Acknowledge clock.

3) Within its PIP Time To Acknowledge clock, Client initiates a new connection to the Service and sends a Receipt Acknowledgement indicating successful receipt of the RosettaNet Business Message. The service acknowledges the Receipt Acknowledgement with an HTTP response code indicating successful receipt.

5.3.4.2 Timers

Timers are useful in measuring time and triggering actions when a specific period of time has elapsed. In this pattern, two measures of time are useful: HTTP response time and PIP Time To Acknowledge.

For each interaction with the service (RosettaNet Business Message pull and Receipt Acknowledgement send), the request and response is on the same connection and the HTTP response time should be no longer than what is a reasonable/acceptable time to leave an HTTP connection open. The duration of the HTTP response time would be agreed upon in the TPA. After such time has elapsed, the client MUST close the connection and treat it as a transport related error.

The PIP Time To Acknowledge maps to the RosettaNet PIP Time To Acknowledgement for the particular RosettaNet Business Message being exchanged. This profile considers the behavior by either the pure client or service when the PIP Time To Acknowledge is exceeded as implementation detail out of scope of this profile.

5.3.4.3 Receipt Acknowledgements

Receipt Acknowledgements exchanged indicate that a RosettaNet Business Message was receiving and it was validated against its schema. They are sent from the recipient to the sender of the message exchange and are used to complete business transactions and support non-repudiation of receipt. The Receipt Acknowledgement in the pure client to service request response pull pattern is sent by the recipient (Pure Client) to the sender (Service) using a subsequent message sent by the Pure Client after successfully receiving a RosettaNet Business Message. This profile considers the behavior of the service after its PIP Time To Acknowledge clock has elapsed as implementation detail and is out of scope of this profile.

5.4 QoS and Service to Service Pattern

This section addresses Web services QoS aspects which are specifically relevant to the pattern involving a Service invoking a Service.

In order to provide comprehensive examples for QoS aspects of the pattern in this section, we define two Services, A and B. ServiceA acts as the initially invoking service, while ServiceB is the initially invoked service. It outlines the QoS aspects of using an RM sequence and which security protection mechanisms are required within the information exchange.

5.4.1 Service To Service One Way Callback Pattern

This section addresses QoS aspects common to patterns with Service To Services. It outlines the QoS aspects of using an RM sequence and which security protection mechanisms are required within the information exchange.

5.4.1.1 Security

Both the invoking and invoked services in this pattern are assumed to be deployed into an enterprise-level environment with robust IT functionality and involved entities where a brokered authentication model can be leveraged, avoiding the limitations of transport-level authentication mechanisms and the credential management tasks of a direct authentication model.

R2036 – Services of these patterns must utilize a security policy containing a sp:AsymmetricBinding element.

5.4.1.2 Reliability

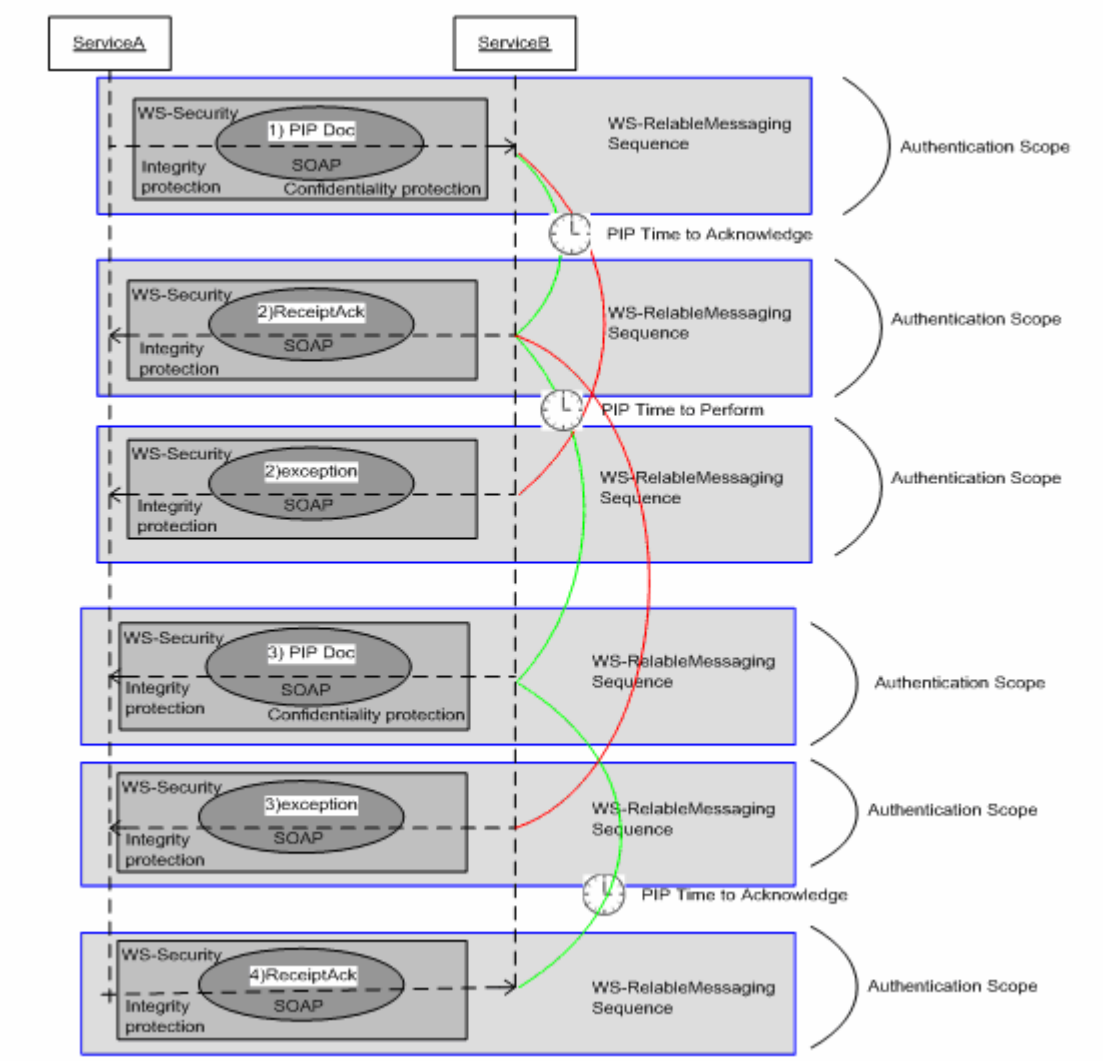
R2037 – WS-ReliableMessaging must be used by services in these patterns. All steps within this pattern must use ExactlyOnce and InOrder delivery assurances within the RM sequence.

5.4.1.3 Timers

Timers are useful in measuring time and triggering actions when a specific period of time has elapsed. In these patterns, the duration of the PIP timers and the resulting actions if timer limits are reached would be agreed upon in the TPA.

5.4.1.4 Overall Flow with QoS

The following diagram defines the flow associated with this pattern.



In the following sequence a WS-ReliableMessaging Sequence is utilized during each exchange and integrity protection is always required.

1) Confidentiality protection is required in this exchange. ServiceA invokes ServiceB's operation accepting a RosettaNet Business Message. At completion of sending the RosettaNet Business Message by ServiceA, ServiceA starts its Time to Acknowledge clock. At acceptance of the RosettaNet Business Message, ServiceB starts its Time to Acknowledge clock.

2) In the case of no errors, within its Time to Acknowledge clock limit, ServiceB invokes ServiceA's operation accepting the RosettaNet Receipt Acknowledgement document. At completion of sending the Receipt Acknowledgement document, ServiceB starts it Time to Perform clock. At

acceptance of the Receipt Acknowledgement document, ServiceA starts its Time to Perform clock.

In the case of errors, within its Time to Acknowledge clock limit, ServiceB invokes ServiceA's exceptionOp operation and the sequence is ended.

3) ***(this text is non-normative***) At the completion of sending the ReceiptAcknowledgement, ServiceB starts its Time to Acknowledge clock. At acceptance of the ReceiptAcknowledgement, ServiceA starts its Time to Acknowledge clock.

   In the case of no errors, confidentiality protection is required. Within its Time to Perform clock limit, ServiceB invokes ServiceA's operation which accepts the resulting RosettaNet Business Message.

   In the case of errors, within its Time to Acknowledge clock limit, ServiceB invokes ServiceA's exceptionOp operation and the sequence is ended.

4) ***(this text is non-normative)*** Within its Time to Acknowledge clock limit, ServiceA invokes ServiceB's operation accepting the RosettaNet Business Acknowledgement document.

6 Glossary

Term	Definition
Business Operation	A ***Business Operation*** is a web service operation that has RosettaNet defined business documents as input or output parameters.
Community WSDL	***Community WSDL*** refers to an existing and approved RosettaNet WSDL document that follows the prescribed WSDL mapping rules. It serves as a template WSDL document that trading partners may use in constructing their own RosettaNet-compliant WSDL documents.
Dual Action Business Interaction	***Dual Action Business Interaction*** is a special case of Multiple Action Business Interaction. It is an Interaction between business partners that involves exchange of two RosettaNet business document. For example, Request Purchase Order interaction involves exchange of two RosettaNet business documents i.e. Purchase Order Request from buyer to seller, and PurchaseOrderConfirmation from seller to buyer.
Hosting-Enabled Partner	A ***hosting-enabled partner*** is a business partner that has full Web service capabilities, supporting the full set of the specifications and standards listed earlier in the background section. It can host a Web service, and provide reliable, secure interactions using the relevant Web services specifications. A hosting-enabled business partner cannot invoke a pure-client.
Multiple Action Business Interaction	***Multiple Action Business Interaction*** is an Interaction between business partners that involves exchange of more than one RosettaNet business document. For example, procurement process could involves exchange of several RosettaNet business documents e.g. Purchase Order Request from buyer to seller, PurchaseOrderConfirmation from seller to buyer, PurchaseOrderStatusNotification from seller to buyer etc.
Pure Client	A ***pure client*** does not host services and cannot be invoked itself as a service. It can have varying Web service capabilities supporting at least the minimal set of specifications and standards listed earlier in the background section.
Role Type	***Role Type*** is the type of role that performs activities in an e-Business process e.g. Sold to, Sold by, Distributed by etc. Valid PIP Roles are specified in the PIP description provided by RosettaNet (e.g. DescriptionOfPartnerInterfaceProcessFor3A4.doc)
Single Action Business Interaction	***Single Action Business Interaction*** is an Interaction between business partners that involves exchange of a single RosettaNet business document. For example, to Distribute Order Status, seller sends a single RosettaNet business document (i.e. *PurchaseOrderStatusNotification)* to buyer.
Signal Operation	A ***Signal Operation*** is used to refer to following two operations: ReceiptAcknowledgmentOp, and ExceptionOp

Signal Schema	A ***Signal Schema*** is used to refer to following three schemas: ReceiptAcknowledgment_00_01.xsd, WS_Exception_00_01.xsd, and WS_GetMessage_00_01.xsd. These schemas are located in ./system/ sub directory.
Trading Partner Agreement (TPA	***Trading Partner Agreement (TPA)*** is an agreement by two or more business partners. When this Profile does not specify something or only provides guidance, the TPA should be used to capture the integration requirements.

7 Appendix A: References

Source	Description
[MCC]	Title: Message Choreography and Control RosettaNet Retrieved July 29, 2008 from: http://members.rosettanet.org/Standards/RosettaNetPrograms/FoundationalPrograms/FormingFoundationalPrograms/MessageControlChoreography/tabid/3096/Default.aspx
[MTOM]	Title: SOAP Message Transmission Optimization Mechanism W3C Recommendation 25 January 2005 Retrieved June 20, 2008 from: http://www.w3.org/TR/soap12-mtom/
[RAE]	Title: RosettaNet Automated Enablement RosettaNet Retrieved June 20, 2008 from: http://www.rosettanet.org/cms/sites/RosettaNet/Standards/Programs/milestone/completed/rae/index.html
[RFC2119]	Title : Key words for use in RFCs to Indicate Requirement Levels The Internet Engineering Task Force Retrieved June 20, 2008 from: http://www.ietf.org/rfc/rfc2119.txt
[SOAP11]	Title: Simple Object Access Protocol (SOAP) 1.1 World Wide Web Consortium (W3C) Retrieved June 20, 2008 from: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
[SOAP12]	Title: SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) World Wide Web Consortium (W3C) Retrieved June 20, 2008 from: http://www.w3.org/TR/soap12/
[SSL]	Title: The SSL Protocol Version 3.0 Netscape Communications Retrieved June 20, 2008 from: http://wp.netscape.com/eng/ssl3/draft302.txt
[TLS]	Title: The TLS Protocol Version 1.0 The Internet Society and The Internet Engineering Task Force Retrieved June 20, 2008 from: http://www.ietf.org/rfc/rfc2246
[TPIR-PIP]	Title: TPIR-PIP® Core Specification RosettaNet Retrieved June 20, 2008 from: http://www.rosettanet.org/cms/sites/RosettaNet/Standards/RStandards/tpir/index.html
[WSAddressing]	Title: Web Services Addressing 1.0 - Core World Wide Web Consortium (W3C) Retrieved June 20, 2008 from: http://www.w3.org/TR/ws-addr-core/
[WSDL]	Title: Web Services Description Language (WSDL) 1.1 World Wide Web Consortium (W3C) Retrieved June 20, 2008 from: http://www.w3.org/TR/wsdl

[WSI]	Web Services Interoperability Organization Retrieved June 20, 2008 from: http://www.ws-i.org/
[WSIBP]	Title:Basic Profile Version 1.1 Web Services Interoperability Organization Retrieved June 20, 2008 from: http://www.ws-i.org/Profiles/BasicProfile-1.1.html
[WSIBSP]	Title: Basic Security Profile Version 1.0 Web Services Interoperability Organization Retrieved June 20, 2008 from: http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html
[WSRM]	Title:Web Services Reliable Messaging (WS-ReliableMessaging) OASIS Retrieved July 02, 2008 from: http://docs.oasis-open.org/ws-rx/wsrm/200702
[WSSSOAP]	Title: Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS Retrieved June 20, 2008 from: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[WSSUP]	Title: Web Services Security: UsernameToken Profile 1.0 OASIS Retrieved June 20, 2008 from: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
[WSSX509P]	Title: Web Services Security: X.509 Certificate Token Profile ~~1.0~~ OASIS Retrieved June 20, 2008 from: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
[WSPolicy]	Title: WS-Security Policy 1.2 OASIS Retrieved July 02, 2008 from: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702
[WSSecurityPolicy]	Title: Web Services Policy 1.5 - Framework W3C Retrieved July 02, 2008 from: http://www.w3.org/TR/2007/REC-ws-policy-20070904/
[WSPolicyAttachment]	Title: Web Services Policy 1.5 - Attachment W3C Retrieved July 02, 2008 from: http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/
[WSRMPolicy]	Title: Web Services Reliable Messaging Policy Assertion (WS-RM Policy) OASIS Retrieved July 02, 2008 from: http://docs.oasis-open.org/ws-rx/wsrmp/200702
[XML]	Title: W3C Extensible Markup Language 1.0 (Third Edition) World Wide Web Consortium (W3C) Retrieved June 20, 2008 from: http://www.w3.org/TR/REC-xml/
[XMLSchemaLanguage]	Editors : Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn

	Title : “XML Schema Part 1: Structures” World Wide Web Consortium Retrieved October 20, 2003 from: http://www.w3.org/TR/xmlschema-1/ Editors : Paul V. Biron, Ashok Malhotra Title : “XML Schema Part 2: Datatypes” World Wide Web Consortium Retrieved October 20, 2003 from: http://www.w3.org/TR/xmlschema-2/

8 Appendix B: Example Use Cases

8.1 Service to Service

8.1.1 Service to Service One Way Callback Use Cases

8.1.1.1 PIP 3A4: Request Purchase Order

1. Within the Buyer's private process, Buyer enters rest of purchase order information (shipping, PO # etc.) in Buyer application.
2. Buyer presses send purchase order button.
3. Buyer application triggers the sending of a one way soap message containing purchase order request XML to the Seller Web service
4. Initiating connection closes without response
5. Seller Web service performs schema validation.
6. If schema validation succeeds: Seller Web service stores purchase order. It then sends a one way soap message containing purchase order confirmation XML, with pending status, to the Buyer Service. Buyer service sends it to the Buyer's private process where it processes the purchase order confirmation, and updates purchase order status to pending.

8.2 Pure Client to Service

8.2.1 Pure Client to Service Request Response Use Cases

8.2.1.1 PIP 3A5: Query Order Status

1. Buyer enters purchase order sales order # in Buyer application.
2. Buyer application queries Seller Web services for order status of the given sales order #
3. Seller Web service performs schema validation, looks up order status, and returns order status XML response synchronously as part of HTTP response.
4. Buyer application displays order status.

8.2.1.2 PIP 3A2: Request Price and Availability

1. Buyer enters product names in Buyer application (e.g. custom or packaged application).
2. Buyer application queries Seller Web services for prices and availability.
3. Seller Web service performs schema validation, looks up pricing and availability, and returns pricing and availability XML response synchronously as part of HTTP response.
4. Buyer application displays pricing and availability information.

8.2.1.3 PIP 3A1: Request Quote

1. Buyer enters rest of information to request quote (billing, and contract information etc.) in Buyer application.
2. Buyer application requests quote from Seller Web services.
3. Seller Web service performs schema validation, creates quote, and returns quote XML response synchronously as part of HTTP response.
4. Buyer application converts the quote to a purchase order.

8.2.2 Pure Client to Service Request Response Push Use Cases

8.2.2.1 PIP 3A7: Notify Of Purchase Order Update

After reviewing the buyer's PO (3A4R) or PO change (3A8R), the seller responds to the buyer with the Notify of PO update (3A7).

1. Seller pushes Notify of Purchase Order Update (3A7) message to the Buyer service.
2. Buyer performs schema validation and can optionally perform additional validations of the message as needed.
3. If (schema) validation succeeds: Buyer stores PO Update and then Buyer returns an acknowledgement in the HTTP response to the seller confirming the receipt/acceptance of the message.

8.2.2.2 PIP 3C3: Notify of Invoice

After seller satisfies all or part of the PO

1. Seller pushes Notify of Invoice message to the Buyer service.
2. Buyer performs schema validation and can optionally perform additional validations of the message as needed.
3. If (schema) validation succeeds: Buyer stores Invoice and then Buyer returns an acknowledgement in HTTP response to the seller confirming the receipt/acceptance of the message.

8.2.2.3 PIP 3A4: Request Purchase Order

1. Buyer enters rest of purchase order information (shipping, PO # etc.) in Buyer application.
2. Buyer presses send purchase order button.
3. Buyer application sends purchase order request XML to Seller Web service.
4. Seller Web service performs schema validation.
5. If schema validation succeeds: Seller Web service stores purchase order. It returns purchase order confirmation XML, with pending status, in HTTP response and then Buyer application processes the purchase order confirmation, and updates purchase order status to pending.

8.2.3 Pure Client to Service Request Response Pull Use Cases

8.2.3.1 PIP 3C4: Notify of Invoice Reject

1. Out of band, the Buyer rejects the invoice received through their ERP system and generates a RosettaNet compliant XML Notify of Invoice Reject message.
2. Seller application pulls Buyer Web service for the Notify Of Invoice Reject message
3. Seller performs schema validation on returned invoice reject message.
4. If schema validation succeeds: Seller stores Notify Of Invoice Reject message and then Seller pushes an acknowledgement message to the Buyer service confirming the receipt of the Notify of Invoice Reject message. The Buyer will update their systems with the notation that the Notify of Invoice Reject was acknowledged by the Seller. The pushed acknowledgement message from the seller is not "acked" and nothing is returned in the transport response.

8.2.3.2 PIP 3B2: Advanced Ship Notification

When the seller is ready to ship one or more line items from a PO or updated PO, they notify buyer with an Advanced Ship Notification (3B2)

1. Seller pushes Advanced Ship Notification message to the Buyer service.
2. Buyer performs schema validation and can optionally perform additional validations of the message as needed.
3. If (schema) validation succeeds: Buyer stores ASN and then Buyer returns an acknowledgement in HTTP response to the seller confirming the receipt/acceptance of the message.

8.2.3.3 PIP 3A4: Request Purchase Order

1. Out of band, the Buyer initiates a purchase order through their ERP system and generates a RosettaNet compliant XML PO Request.
2. Seller application pulls Buyer Web service for purchase order request
3. Seller performs schema validation on returned purchase order request.
4. If schema validation succeeds: Seller stores purchase order and then Seller pushes an acknowledgement message to the Buyer service confirming the receipt of the PO. The Buyer will update their systems with a pending PO status. The pushed acknowledgement message from the seller is not "acked" and nothing is returned in the transport response.

8.2.3.4 PIP 3A8: Request Purchase Order Change

1. Out of band, the Buyer initiates a purchase order update through their ERP system and generates a RosettaNet compliant XML PO Change Request.
2. Seller application pulls Buyer Web service for purchase order change request
3. Seller performs schema validation.
4. If schema validation succeeds: Seller stores purchase order change request and then Seller pushes an acknowledgement message to the Buyer service confirming the receipt of the PO change. The Buyer will update their systems with a pending

status for the PO change request. The pushed acknowledgement message from the seller is not "acked" and nothing is returned in the transport response.

9 Appendix C: Schemas Specific To This Profile

The following schemas can be found inside the profile package. Receipt Acknowledgment and Exception schema are required since MMS Web Services only uses schema (and not DTDs)

- XML\System\ ReceiptAcknowledgment_00_01.xsd
- XML\System\WS_Exception_00_01.xsd
- XML\System\CodeList\WS_MessageError_00_01.xsd
- XML\System\WS_GetMessage_00_01.xsd

9.1 ReceiptAcknowledgment_00_01.xsd

The schema is created by converting RNIF Receipt Acknowledgment DTD to XML Schema

9.2 WS_Exception_00_01.xsd

The schema is created by converting RNIF Exception DTD to XML Schema. Note the following information which cannot be enforced within the schema.

There are five Error Codes which are further classified as one of the two Exception Types - GEE (General Error), or RAE (Receipt Acknowledgement Error).

The Exception\Type element of the WS_Exception_00_01.xsd is used to specify the Exception type. Exception\Description\Error element is used to specify the Error code

9.2.1 GEE (General Error)

General.Error: This is catch all. If no other errors are appropriate to the implement, this error can be raised. This is similar to RosettaNet defined error: PRF.ACTN.GENERR: Error during action performance
Business.Rule.Error: This should be used for any error encountered while validating against any custom business rule. This is not for schema validation errors.
Content.Not.Available: It is generated for Pure Client Request Response Pull MEP when the service does not find response content.
Duplicate.MessageID: When WS-Addressing:MessageID is used in retries in the pure client scenarios duplicate messages can be detected. If duplicate message is detected by the service, then it uses this error code.

9.2.2 RAE (Receipt Acknowledgement Error)

Schema.Validation.Error: This is used when the schema validation failed. This is similar to RosettaNet defined error: UNP.SCON.VALERR: Error during unpackaging – Validating the Service Content.

9.3 WS_MessageError_00_01.xsd

This defines the error codes for the exception schema.

9.4 WS_GetMessage_00_01.xsd

This schema is used in pure client to service request response pull. It is the template input schema for the MEP.