

Message Control and Choreography (MCC)

-

Profile-Web Services (WS)

Release 11.00.00A

Specification Information	
Name	MCC – Profile- Web Services
Publication Date	1 June 2010
Version Identifier	Release 11.00.00A

Table of Content

1.	Document Management	4
1.1	Legal Disclaimer.....	4
1.2	Copyright.....	4
1.3	Trademarks.....	4
1.4	Acknowledgments	5
1.5	Related Documents	5
1.6	Document Version History	5
1.7	Document Purpose	5
2.	Introduction.....	6
3.	Web Service-Based B2Bi	7
3.1	Abstract and Concrete WSDL Definition	8
3.2	Point-2-Point Communication.....	9
3.3	Protocol Layering	10
3.4	WS-* standards	11
3.5	WS-I	11
3.6	WS-BPEL	11
3.7	DTD-based PIPs.....	12
4.	Context of this specification.....	14
4.1	Relation to Single Business Document PIP Template.....	14
4.2	Relation to MMS WS profile.....	15
5.	Single Business Document PIP Definition and Requirements.....	17
6.	Concrete Requirements Defined in the Template Document	18
6.1	Functional Requirements	18
6.1.1	Parties Involved	18

6.1.2	Business Document.....	18
6.1.3	Business State Alignment Features.....	18
6.1.4	PIP Execution Outcome	20
6.2	B2Bi Quality of Service Features.....	20
6.3	Other Non-functional Requirements	21
7.	PIP Parameterization and Execution Control.....	22
7.1	PIP Property Parameters	23
7.2	Further PIP execution parameters.....	24
8.	Realization.....	26
8.1	Execution Contexts and MEPs	26
8.2	Realization of QoS (Strict execution context).....	26
8.3	Control flow (Strict execution context)	29
8.3.1	Asynchronous Interaction	30
8.3.2	Synchronous Interaction.....	34
8.3.3	PIP Instance Identification and Message Correlation	36
9.	WSDL Mapping Rules.....	40
9.1	Messages.....	40
9.1.1	Importing Message Types	40
9.1.2	Defining WSDL Messages.....	41
9.2	Operations	43
9.2.1	Operations Required for Mapping Message Exchange Patterns.....	46
10.	Use Cases of PIP Definition (Strict execution context)	51
10.1	Use Case 1 – Full features	51
10.2	Use Case 2 – Business Document Only	55
11.	Use Case realization (Strict execution context).....	58

1. Document Management

1.1 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.

1.2 Copyright

©2010 RosettaNet. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the inclusion of this copyright notice. Any derivative works must cite the copyright notice. Any public redistribution or sale of this publication or derivative works requires prior written permission of the publisher.

1.3 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.

1.4 Acknowledgments

This document has been prepared by RosettaNet (<http://www.rosettanet.org/>) from requirements gathered during the Milestone Program and in conformance with the methodology. Listed below are the legal entities that contributed to the design and development of this PIP.

Axway	Cisco
DHL	IBM
KJC Solutions	Oracle
OASIS	Software AG
Tibco	University Bamberg
Vienna University of Technology	

1.5 Related Documents

- MCC Single Business Document PIP Template R11.00.00A
- MMS Web Services Profile V11.00.01

1.6 Document Version History

<u>Version</u>	<u>Date</u>	<u>Description</u>
Release 11.00.00A	1 June 2010	Released Version

1.7 Document Purpose

The purpose of the document is to explain the structure, the association between objects, the content of objects and the definition for single elements to a non-technical audience.

2. Introduction

Message Control and Choreography (MCC) phase 1 specifies the execution of 1-Action (single business document) PIPs using standard messaging technologies, thereby providing an alternative to using the RosettaNet Implementation Framework (RNIF). The *"MCC Phase 1 – Single Business Document PIP Template"* (the *template document* in the following) motivates the use of standard messaging technologies, sets the scope for MCC phase 1, and defines common criteria for messaging technology profiles.

This document specifies how to use Web services for performing 1-Action PIPs. Section 3 describes elementary characteristics of Web service based Business-to-Business integration (B2Bi) that influenced the development of this specification and set the background for this specification. Section 4 describes the relation of MCC phase 1 WS profile to *"MCC Phase 1 – Single Business Document PIP Template"* and to what extent the existing *"RosettaNet MMS WS profile"* is reused. Section 5, which is directly taken from *the template document*, defines core MCC concepts. Section 6 repeats the requirements defined in *the template document* and section 7 describes how PIPs can be parameterized. Section 8 specifies the realization of PIPs. Section 9 defines how this specification's contents map to WSDL interface descriptions. Section 10 recites use case definitions from *the template document* and section 11 discusses the corresponding implementation.

3. Web Service-Based B2Bi

Web services are an XML-based interface technology that separates the functionality of a service from its implementation. A Web service's interface is described using the Web service description language (WSDL). A WSDL interface of a service defines the requirements for the so-called *service provider* (that implements the WSDL interface) and the assurances a service consumer (that uses a WSDL interface implementation) can rely on. Thus, Web services technology specifies what happens during communication *between* service provider and service consumer and leaves out the details of internal processing of the communication partners. Web services enable decoupling of communication partners by providing a concise description of all relevant communication requirements. Leveraging this potential for decoupling allows for choosing platforms and implementation languages for service consumers and providers independently.

This document is aligned with the goal of decoupling PIP implementers' information systems as far as possible and allowing for reuse of PIP implementations. Instead of specifying the details of internal behavior of RosettaNet implementers, the MCC phase 1 Web services profile only defines requirements for communication.

Some of these requirements concern the realization of B2Bi over the Internet which was not reflected sufficiently in the early Web service specifications, most notably, stateful interactions and Quality-of-Service (QoS). The use of WS-BPEL as major technology for implementing stateful interactions and various WS-* standards for realizing QoS is assumed, but not mandated.

Therefore, this specification does rely on basic Web service specifications like WSDL and SOAP, Web service extensions like WS-ReliableMessaging and WS-Security, orchestration technology (WS-BPEL) as well as interoperability definitions like WS-I Basic profile and WS-I Reliable Secure Profile.

3.1 Abstract and Concrete WSDL Definition

*"A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service."* (WSDL 1.1)

An abstract WSDL definition that is not tied to a particular service implementation comprises the following components:

- Type definitions using XML Schema technology.
- Message definitions for composing messages from types.
- PortType definitions for grouping the exchange of messages in operations.

A concrete WSDL definition extends an abstract WSDL definition by the following components:

- A Protocol Binding definition that describes how the messages of a portType's operations are exchanged using a concrete protocol.
- Port definitions for specifying which network addresses can be used for consuming a service.
- A Service definition for bundling one or more ports for the same service.

Port and Service definitions have to be specified by Rosettanet implementers on a per PIP basis. The MCC phase 1 WS profile therefore does not make any assumptions about Port and Service definitions and only defines requirements for protocol bindings. Conversely, this specification makes strict requirements for the components of abstract WSDL definitions for performing PIPs.

Thus, this specification defines abstract requirements that then can be tailored to the specific systems environment of RosettaNet implementers.

<i>This profile does not specify details of WSDL Port, Service or Protocol Binding definitions. Protocol Bindings are constrained by requirements.</i>
--

3.2 Point-2-Point Communication

Consuming a Web service requires knowledge (statically or at runtime) of an endpoint that provides the implementation of an abstract service description. At the application level, communication directly takes place between the *service consumer* and the *service provider*. This amounts to a Point-2-Point communication model as depicted in Figure 1.

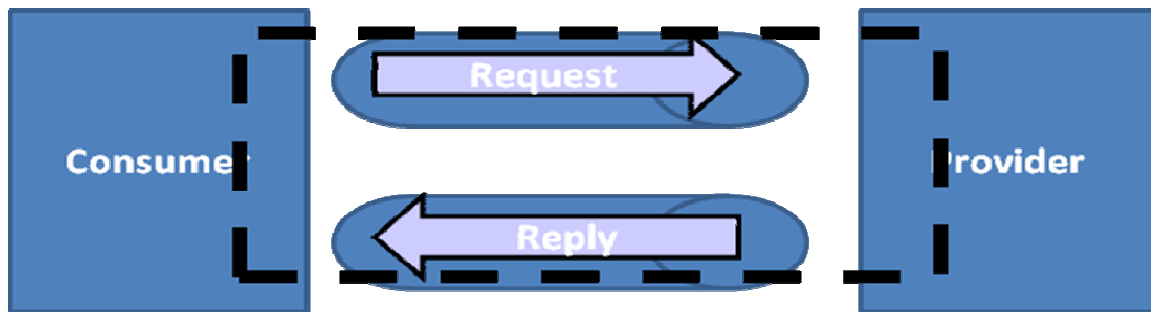


Figure 1: Point-to-Point Communication

In a Point-2-Point model QoS requirements may be implemented on the communication channel, e.g., using WS-Security, as well as by the communicating applications, e.g., using digital signatures contained in the payload. The MCC phase 1 WS specification assumes a Point-2-Point communication model.

Therefore, if two partners decide to apply an End-2-End communication model with intermediaries as depicted in Figure 2, then the implications of Point-2-Point communication have to be adapted accordingly. In particular, this specification's requirements for communication channel level QoS realization have to be adapted.

This profile supports Point-2-Point communication only.

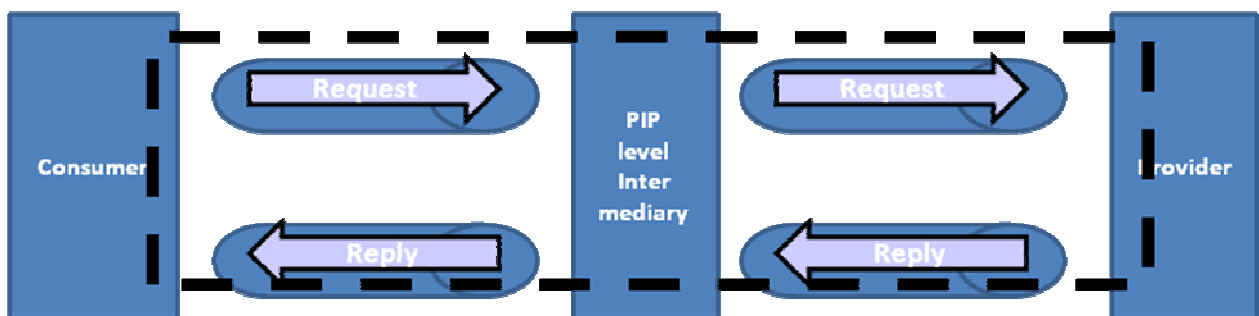


Figure 2: End-to-End Communication

3.3 Protocol Layering

Web service-based communication allows for different types of protocols. For full conformance with this specification, Web services must be bound to SOAP via HTTP. This results in the protocol stack depicted in Figure 3.

This profile mandates the use of SOAP via HTTP for Web service communication.



Figure 3: Protocol Stack for WS-based PIP Implementation

The highest layer is the PIP Protocol, and specifies the sequence of Web service calls that are needed for implementing a PIP. This may concern WS calls needed for alignment of business state as well as control messages signaling protocol exceptions like timeouts. Each Web service operation is bound to SOAP communication and may result in a sequence of SOAP level messages depending on the configuration of the SOAP layer. For example, reliable delivery of PIP level messages may require the exchange of several SOAP level messages. Each SOAP message will be exchanged using HTTP which in turn has to be bound to TCP/IP as transport protocol. In the subsequent sections, messages and functionalities may be associated with the layers of the WS-based PIP protocol stack depicted in Figure 3.

- A message is said to be a PIP level message if it is defined as input or output of a PIP WSDL operation. Functionalities are said to be realized at the PIP level if they are realized by the protocol machines implementing the PIP protocol. The PIP level corresponds to what is denoted the “application level” in general.
- A message is said to be a SOAP level message if its format conforms to the SOAP specification. SOAP level messages may be exchanged for transmitting PIP level messages as payload and for controlling the exchange of PIP level messages. Functionalities are said to be realized at the SOAP level if they are realized by SOAP processors. The SOAP level corresponds to what is denoted the “messaging level” in general.
- A message is said to be a HTTP/TCP level message if its format conforms to the HTTP/TCP specification. This document does not distinguish between the HTTP and TCP layer which actually are separate. Therefore, HTTP and TCP level messages and functionalities will be referred to as *transport level* messages and functionalities.

While full conformance with this specification requires support for the protocol stack

depicted in Figure 3 RosettaNet implementers are free to adopt different messaging/transport protocols by porting requirements for the respective protocol layers accordingly.

3.4 WS-* standards

WS-* is a non-official, but widely used term denoting Web service related standards that provide advanced features such as QoS. Frequently, these standards define functionality in terms of SOAP-based protocols.

This specification makes use of the following WS-* standards:

- WS-ReliableMessaging 1.2
- WS-Security 1.1
- WS-SecureConversation 1.4
- WS-Trust 1.4
- WS-Addressing 1.0

3.5 WS-I

Interoperability is crucial for reuse of PIP implementations and therefore of paramount importance for the RosettaNet community. The following profiles of the *Web Service Interoperability Organization* therefore must be followed:

- WS-I Basic Profile 1.1 (<http://www.ws-i.org/Profiles/BasicProfile-1.1.html>)
- WS-I Basic Security Profile 1.1 (<http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html>)

Moreover, adherence to the following WS-I specifications is recommended once they become final:

- WS-I Basic Profile 1.2
- WS-I Basic Profile 2.0
- WS-I Reliable Secure Profile 1.0

3.6 WS-BPEL

This specification acknowledges the importance of BPEL as major implementation technology for stateful Web service interactions. Therefore, realization of this specification's requirements using WS-BPEL is a major design goal although RosettaNet implementers are free in choosing a Web service implementation technology.

<i>This profile does NOT require the use of WS-BPEL for implementing PIPs.</i>
--

In order to allow for WS-BPEL based implementations, full conformance to MCC phase 1 WS profile mandates the use of WSDL 1.1.

3.7 DTD-based PIPs

WSDL Web service interface definitions rely on XML schema (XSD) for defining message types. Use of DTD-based PIPs is enabled by equivalent XSD definitions that accompany DTD-based PIPs. Note that these XSD definitions for DTD-based PIPs are purely technical conversions of DTD-based PIPs' monolithic message structure. This is different from the modular message header structure available for XSD-based PIPs labeled with version 11.00.00 and onwards. In order to separate modular XSD PIP definitions from XSD representations of DTD-based PIPs, the term *conversion XSD* will be used in this document to refer to the monolithic equivalent of a DTD PIP definition.

Conversion XSDs define the same set of XML documents as the corresponding DTDs except for the following differences:

- Conversion XSD PIP definitions define an XML namespace for the XML type definitions.
- Conversion XSD PIP definitions may be more restrictive in enumerating the admissible values for PIP code lists.

This enables reuse of business document processing logic with little to no rework. Consider that Web service stack implementations typically do not perform XML schema validation upon operation parameters or at least allow for turning off that feature. Hence, reuse of DTD-based software modules may be as easy as writing a wrapper for the module that exchanges the XML document definition header in business document instances. For example, such a wrapper would replace the XML schema-based XML document definition header by a DTD-based XML document definition header for a business document instance submitted to the DTD-based software module.

For clarification, Figure 4 exemplifies the use of a conversion XSD in a Web service-based integration scenario. In this scenario, the PIP responder (as defined in the *template document*) provides a Web service interface for consuming business documents in its gateway or business service interface. This WSDL interface uses a conversion XSD to import the message type definition of the DTD-based PIP. The WSDL definition is processed by a Web service stack (WS-Stack) implementation for providing the actual service endpoint. This service endpoint can be used by the integration partner for delivering business documents that conform to the type definition expressed in the conversion XSD. This business document is packaged within a SOAP envelope. Whether or not the integration partner used the PIP's DTD or conversion XSD for creating the business document is transparent to the PIP responder. Incoming calls are processed by the provider gateway such that the SOAP message container is stripped off the business document and the actual content is delivered to the PIP responder's message processor that implements the WSDL file. Typically, a WS-Stack does not perform XSD validation upon incoming calls or this functionality can be turned off. Whether or not XML schema validation is performed upon incoming business documents depends on whether or not the message processor performs XSD validation. When receiving a message the message

processor delegates the business content to a DTD-based legacy application that uses the PIP's DTD definitions for type checks. If necessary, the message processor may replace the XSD-based XML document header definition in the business document with a DTD-based XML document header definition.

In this scenario, the integration partners are free to require XSD's advanced validation features or DTD validation. Depending on this choice, the PIP responder may want to perform XSD checks in the message processor component or to use existing DTD based type checks of the DTD-based legacy systems.

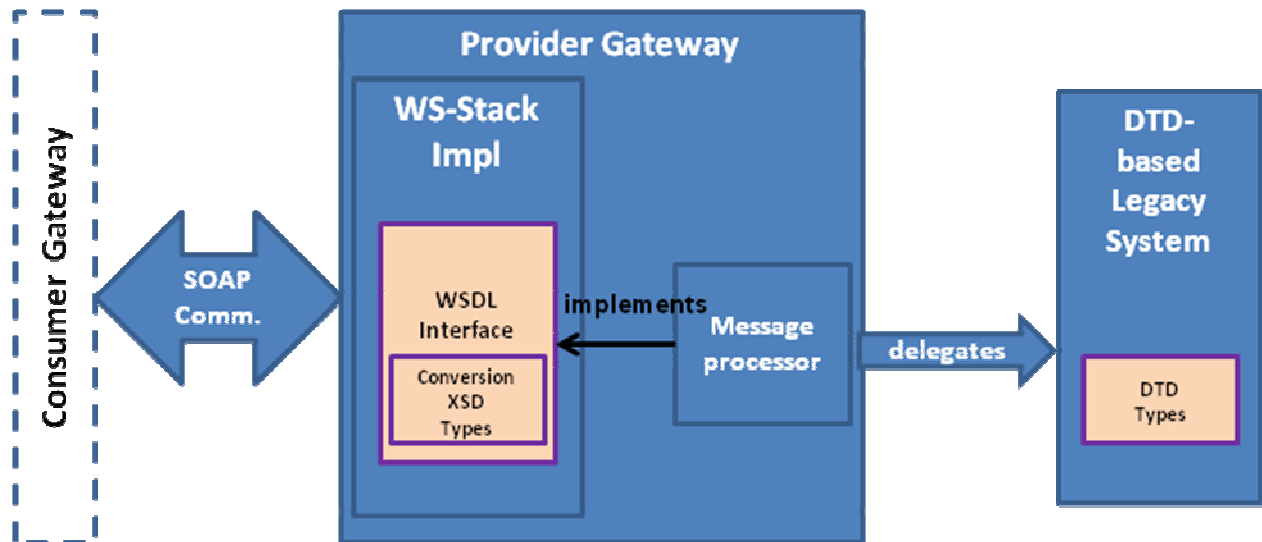


Figure 4: Example use of a Conversion XSD PIP business document definition

In different scenarios, the DTD-based legacy system may also be replaced by an XSD capable business application.

This profile does NOT make assumptions about whether or not conversion XSD-based XML content will be processed by a DTD-based application.

Interoperability considerations require that RosettaNet implementers use the official RosettaNet conversion XSDs for performing DTD-based PIPs with Web service technology. In exceptional cases, RosettaNet implementers may want to derive partner-specific conversion XSDs from DTDs. Although this is a valid approach from a technical point of view, this approach is not recommended.

For full conformance with this profile, officially distributed RosettaNet conversion XSDs MUST be used for performing DTD-based PIPs with Web service technology.

4. Context of this specification

This specification (MCC phase 1 Web services profile) governs the execution of PIPs using Web services technology. It is one of several communication technology profiles that implement the requirements defined in the “*MCC Phase 1 – Single Business Document PIP Template*” (*the template document*) that sets the scope and purpose of MCC phase 1. This scope and purpose is similar to the RosettaNet MMS effort. The next sections clarify the relation of MCC phase 1 WS profile to the template document as well as to the MMS WS profile.

4.1 Relation to Single Business Document PIP Template

The “*MCC Phase 1 – Single Business Document PIP Template*” (*the template document*) motivates the MCC effort’s goals and partition into phases. Put short, MCC phase 1 governs the execution of single PIPs while MCC phase 2 specifies the composition of PIPs and execution of PIP compositions. This distinction is reflected in two different *execution contexts* for performing single PIPs. The “***strict PIP execution***” context reflects the situation that a PIP may be used within a PIP composition which imposes hard requirements with respect to mutual agreement upon the PIP result. Agreement is absolutely necessary for PIP compositions in order to ensure consistent routing, i.e., that integration partners perform and accept the same sequence of PIP executions. The “***lax PIP execution***” context reflects the situation that a PIP is not to be reused within PIP compositions and therefore has less stringent requirements with respect to mutual agreement. In particular, PIP result revisions or error notifications may be acceptable.

A core achievement of MCC phase 1 is configurability of PIPs, i.e., integration partners are free to adjust several performance controls such as PIP timeouts or encryption. To separate between a PIP as defined by the RosettaNet community and partner-customized PIPs, the concepts of ***PIP template***, ***PIP definition***, ***customized PIP*** and ***PIP instance*** are defined. Section 5, which is directly taken from the *template document*, clarifies the meaning of these concepts.

The *template document*’s requirements are explicitly restated here with minimal to no adjustments that reflect Web service specifics:

- The *template document*’s requirements on state alignment features and PIP execution outcome are restated in section 6.1.
- The *template document*’s requirements on QoS are restated in section 0.
- The *template document* requires the use of ebXML BPSS (ebBP) as format for representing *PIP definitions* as well as *customized PIPs*. Section 7.1 describes the available PIP configuration options.
- The PIP execution modes defined in the *template document* are restated in section 7.2. Note that ***PIP execution modes*** are different from ***PIP execution contexts*** in reflecting architectural design parameters of performing single PIPs instead of distinguishing between whether or not a PIP may be reused within a PIP composition.

4.2 Relation to MMS WS profile

MCC phase 1 and MMS are similar in scope. MMS governs the exchange of one business document and MCC phase 1 deals with performing PIPs. Since RosettaNet will only support One-Action-PIPs (single business documents) in the future the difference between MCC phase 1 and MMS is not about the number of business documents to be exchanged. Having composition of BTAs in mind, MCC phase 1 defines different requirements for business document exchanges as well as different assumptions about communicating partners:

1. MCC phase 1 distinguishes between ***lax PIP execution*** and ***strict PIP execution*** where strict execution imposes hard requirements with respect to mutual agreement between integration partners upon PIP result.
MMS does not assume as hard agreement requirements and therefore does not support the strict PIP execution context. This concerns the realization of QoS properties in a mutual way as well as a stringent definition of control flow.
2. MCC phase 1 considers that composition of PIPs (MCC phase 2) is inherently complex. Complexity concerns support for business process instances and relating business messages to these process instances, support for monitoring timing constraints as well as support for validation of business messages. This profile therefore assumes that an MCC capable integration partner provides an adequate system environment for providing these capabilities. This also includes the capability of providing an endpoint for communication.
MMS does not assume as many integration partner capabilities. In particular, MMS' concept of *pure clients* allows for integration partners that cannot offer addressable endpoints. To accommodate pure clients, MMS defines multiple message exchange patterns (MEPs) that leverage the concept message pulling for dealing with non-addressable participants.

The discussion shows that MCC phase 1 and MMS are similar in scope but have different goals. Consequently, MCC requirements and MMS assumptions cannot freely be combined. In particular, implementing the strict PIP execution context requirements on MMS assumptions implies unacceptable complexity for integration partners. The lax PIP execution context, though, is supported by MMS as is.

Therefore, MCC phase 1 WS profile leverages the MMS WS profile for implementing pulling-based MEPs with lax PIP execution context. For the strict PIP execution context, this document specifies how to perform PIPs assuming that both PIP participants provide reasonable system support. Table 1 summarizes the choice between this specification and MMS WS profile depending on execution context and supported MEPs.

	Service-To-Service MEP	Pulling-based MEPs
Strict execution context	MCC phase 1 WS profile	NOT supported
Lax execution context	MCC phase 1 WS profile or MMS WS profile	Use MMS WS profile as is

Table 1: Decision Matrix for Choosing between MCC and MMS

The rest of this document covers the execution of PIPs according to the strict PIP execution context. MMS WS profile concepts will be reused where possible. This concerns conventions such as naming conventions for WSDL contents as well as technology choices such as WS-ReliableMessaging. The key differences in comparison to the MMS WS profile are:

- QoS property realization in a mutual way.
- Stringent control flow definition.
- Message correlation based on PIP service header elements.
- Support for DTD-based PIPs.

5. Single Business Document PIP Definition and Requirements

<p><i>This page is taken from "MCC Phase 1 – Single Business Document PIP Template" AS IS.</i></p>
--

The "Single Business Document Template" section defines a model for single business document PIPs that is aligned with ebBP Single business document BusinessTransactions. It is abstract in two different ways:

1. The realization of a PIP definition component may vary with the communication technology selected for implementing the PIP.
2. The realization of a PIP definition may vary depending on the execution context assumed.

Also, the template for Single Business Document PIP definition is general in the sense that the definition of a concrete PIP will select from the model components offered. Section "Execution Parameters and Configuration" therefore describes rules for defining a customized, or "concrete" PIP.

To summarize, there are four levels at which PIP material is defined:

- (1) **PIP template:** This level defines the general structure – or model - of a PIP and the features that may be used in a particular PIP definition. This is the object of this document.
- (2) **PIP definition:** This level defines particular PIPs usable for business exchanges. These will usually contain parameters that are left for users to define, e.g. via an agreement between members of a supply chain. A PIP definition is prescriptive and states the conditions for a PIP instance to be considered as conforming to a PIP definition.
- (3) **Customized PIP:** (or concrete PIP): At this level, all elements of a PIP are fully defined, and all parameters (such as QoS, timing) are given a specific value or specific range that is agreed upon between partners. The execution of such PIPs is determined in terms of QoS, alignment features and execution mode. The factors that condition a successful or a failed outcome are fully determined and known from partners.
- (4) **PIP instance:** This is an image of a particular execution of a PIP, i.e. a particular sequence of concrete messages where all components and PIP properties are given a value – e.g. a fully defined business document between two identified partners, a particular timing between these messages, etc.

6. Concrete Requirements Defined in the Template Document

This section restates the “*MCC Phase 1 – Single Business Document PIP Template*” (*the template document*) requirements for the MCC phase 1 WS profile. Note that the business state alignment features of section 6.1.3 are abstract. The implementation of alignment is specified in later sections. For example, validity alignment is implemented using ReceiptAcknowledgement messages that are known from RNIF or ebBP.

6.1 Functional Requirements

6.1.1 Parties Involved

Since RosettaNet supports 1-Action PIPs only, there is only one business document to be exchanged when performing a PIP. Any PIP has exactly two participants:

- (a) the **PIP requester** party (or Requester), sending the Single Business Document message.
- (b) the **PIP responder** party (or Responder), receiving the Single Business Document message.

6.1.2 Business Document

Business documents represent the actual business content of the PIP as defined in RosettaNet business document definitions. Business documents may be complemented with additional attachments. In particular, binary content such as drawings or construction plans may be added. Attachments may be subject to processing steps performed upon business documents like validation or storage and therefore may influence the result of PIP executions. The default rule is that business documents and attachments must be treated as a single entity and that all processing steps applied to a business document are applied to attachments as well. The integration partners are free to define different processing rules.

6.1.3 Business State Alignment Features

The objective of these alignment features is to provide to each business party participating to a PIP, a common understanding about the status of the business document in terms of its reception, validity and processing prospects. Two features stand out:

- **(1) Delivery Alignment**
Aligns the information about whether or not the business document has been delivered to the PIP responder. As opposed to the *template document*, this profile does not distinguish between delivery to the messaging layer and delivery to the application layer. It is assumed that the PIP responder will ensure that a message that has successfully been delivered to the messaging

layer will also be delivered to the application layer. This profile does not specify the realization of this assumption. Implementation options range from persistent message logs to synchronous delivery.

- **(2) Validity Alignment**

Aligns the information about whether or not the PIP responder considers the business document to be valid. This profile acknowledges that the concept of validity and its technical and legal implications rely on the specifics of PIP implementers. Therefore, the exact meaning of validity as well the validation steps to be performed before alignment are to be specified by the integration partners. The following four validation steps are defined as default (may be overridden by implementers):

- **Syntax validation**, i.e., check whether the business document is a well-formed document.
- **Type validation**, i.e., check whether the business document is valid according to a schema definition file.
- **Business Rules validation**, i.e., check whether the business document is in line with a set of business rules that can be automatically checked without touching business applications.
- **Sequence validation**, i.e., check whether this kind of business document is expected at the current state of the super-ordinate collaboration (if any).

The following list of additional steps as well as non-listed validation procedures may be defined by PIP implementers:

Additional steps (non-comprehensive)

- **Business entity dereferencing**, i.e., check whether the business entities defined in the business document can be resolved within the business application.
- **Document completeness check**, i.e., check whether the business document is complete from a business perspective. This may concern completeness of line items as defined in a business document of a prior PIP or as required by a business application.
- **Business application check**, i.e., the responder party must make sure that any validation checks have been applied to the action message that are necessary for ensuring processability of the business message.
- **Delegation to business application**, i.e., the business document has successfully been imported by the business application.

6.1.4 PIP Execution Outcome

A Single Business Document PIP result is defined as a Protocol-outcome:

- **Protocol-outcome** is a label of value in {SUCCESS, FAILURE} where:
 - SUCCESS means: The PIP execution can be considered as fully conforming to the PIP definition or to the concrete PIP: alignment requirements, QoS requirements and other execution mode requirements have been satisfied.
 - FAILURE means: The PIP execution has been deficient in some way and violated some requirements in the PIP definition or the concrete PIP: alignment requirements, QoS requirements and other execution mode requirements, have not been observed.

The protocol-outcome is technical and business-outcome may deviate. Whether or not a SUCCESS protocol-outcome is also a business success may depend on the evaluation of the business content that has been exchanged. Conversely, a FAILURE protocol-outcome cannot coincide with a business success.

Note that, as opposed to the *template document* definition, the strict PIP execution context requires that both PIP requester and PIP responder know the PIP protocol outcome at the end of the PIP protocol.

*The **strict PIP execution context** requires the PIP protocol-outcome to be known by both PIP requester and PIP responder immediately after the PIP execution.*

6.2 B2Bi Quality of Service Features

B2Bi has special requirements concerning the realization of QoS. Security and reliability are of paramount importance. The realization of QoS properties does not only concern the message that is exchanged but also whether PIP requester and PIP responder know that it has been realized. As there are two roles, QoS properties may be realized asymmetrically. For example, by attaching a signature to a message the receiver can verify the authenticity of the sender but the sender cannot verify the authenticity of the receiver. A QoS feature is said to be realized in a mutual way if it is implemented for both roles of a PIP, i.e., PIP requester and PIP responder. Whether or not asymmetric or mutual implementation of QoS properties is acceptable is a core difference between the lax PIP execution context and the strict PIP execution context. The implementation of QoS properties in a mutual way is discussed in section 8.2.

(1) Security options

- Authentication
- Confidentiality

- Integrity
- Authorization
- Non Repudiation/Non Repudiation Of Receipt

(2) Reliable Messaging

- Guaranteed delivery (At-least-Once delivery)
- Duplicate elimination (At-Most-Once delivery)
- Exactly Once delivery

(3) Timing Constraints

- **Time to acknowledge validity (or invalidity)**
Measured from the time the business document has been delivered until alignment of validity has been performed. Note that the integration partners are free to define the notion of validity as pointed out in section 6.1.3 ("validity alignment").
- **Time to Perform**
Measured from the start of a PIP until the last alignment message has been exchanged. Depending on whether or not a PIP is initiated by a super-ordinate process instance the start may coincide with a super-ordinate initialization message or with the delivery of the business document.

6.3 Other Non-functional Requirements

Section 6.2 defines QoS environments from a B2Bi point of view. Traditionally, QoS is defined to cover measurable network qualities like latency or throughput. However, such requirements explicitly are not subject to MCC investigation.

7. PIP Parameterization and Execution Control

Allowing for customized PIPs is a core design goal of MCC phase 1. The ebBP format that allows for abstract specification of QoS parameters has been chosen for representing the customized PIPs. Customization options are referred to as PIP property parameters and can easily be shared across the community. For concrete implementation, more parameters have to be specified that cannot easily be shared. These parameters are referred to as PIP execution parameters. The *template document* describes the difference of between property parameters and execution parameters as follows.

1. **PIP property parameters:** These are parameters that control the use of features defined above as PIP properties: level of state alignment and various QoS features. A concrete PIP definition may impose some values / settings for some, and leave some values open or within a range, for others. A default or recommended value may be suggested, with each concrete PIP definition.
2. **PIP execution parameters:** These are parameters that control the actual execution of the PIP. Most of these will be specific to the messaging solution in use, but some will be defined here independently from these messaging solutions. Indeed, such parameters may help harmonize a PIP usage across messaging solutions.

7.1 PIP Property Parameters

The following parameters are configurable on a PIP definition and a customized PIP basis. The specification items reflect ebBP's abstract specification options for PIPs. The implementation of the specification options is described in section 8. Example ebBP models of PIP property parameters are given in section 10.

Specification item	Configurable	Implication	Explanation
Send Request Document	no	-	A request document always has to be sent.
Overall Time-To-Perform	yes	-	Time for performing the PIP protocol.
ReceiptAcknowledgement	yes	-	May be used for implementing alignment of validity.
Non-Repudiation	yes	-	--
Non-Repudiation-of-Receipt	yes	Sending a ReceiptAcknowledgement	--
TimeToAcknowledgeReceipt	yes	Sending a ReceiptAcknowledgement	Time for sending a ReceiptAcknowledgement measured from the receipt of the action message. If this property is not specified or set to null and if, at the same time, the use of a ReceiptAcknowledgement is required then the PIP requester potentially must wait forever.
Reliability	yes	Turning off reliability implicates "lax execution context"	The "strict execution context" (cf. section 4.2) requires mutual agreement regarding delivery of messages. This, in turn, requires the use of a reliable messaging facility.
Confidentiality	yes	-	--
Integrity	yes	-	--
Authentication	yes	-	--
Authorization	yes	-	--
IntelligibleCheckRequired	yes	Sending a ReceiptAcknowledgement	If set to true, implies the default validation procedures defined in section 6.1.3 or the validation steps defined by the integration partners. If set to false, implies that no validation steps will be performed. A ReceiptAcknowledgement may be exchanged nonetheless.
RetryCount	yes	-	Describes how often a business document/signal is to be resent at the PIP process level.

Table 2: PIP Property Parameters

7.2 Further PIP execution parameters

Message Exchange Patterns: These are usually conditioned by connectivity constraints. These exchange patterns may affect the way QoS is achieved as well as state alignment. Three MEPs are defined here that are expected to cover most execution cases, but are not exclusive of others. Some may only be applicable to some transport protocols. These MEPs, however, are defined abstractly from these transports while defining some invariant properties across these transports:

- **Synchronous execution** (requester-initiated): The action message is pushed from the requester to the responder party, while any form of receipt (implementing some state alignment feature) is sent back over the same connection synchronously. This MEP only applies for request-response transports such as HTTP, where receipts can be sent over the response leg.
- **Asynchronous execution with callback** (requester-initiated): The action message is pushed from the requester to the responder party, while any form of receipt (implementing some state alignment feature) is sent back as a callback asynchronously on a different connection. This MEP is appropriate when the timing for producing the receipt prohibits using the same connection. **Invariants:** This MEP assumes addressability of both requester and responder, and readiness to receive incoming messages.
- **Asynchronous execution with pulling** (requester-initiated): The action message is pushed from the requester to the responder party, while any form of receipt (implementing some state alignment feature) is sent back asynchronously as result of a later message pull from the requester. This MEP is appropriate when the timing for producing the receipt prohibits using the same connection and the initiator (requester) is not addressable. **Invariants:** This MEP assumes that the requester takes the initiative of receiving the receipt: it is using a request-response exchange (with receipt over the response) for request-response transports such as HTTP. For another non-request-response protocol such as email, the receipt may be pushed first (e.g. SMTP) to some intermediate store, then pulled by the requester (e.g. using a client protocol such as IMAP).

Protocol Binding:

Protocol Binding is covered by MMS WS profile V11.00.00, section 4.3.

Support for attachments:

Support for attachments is covered by MMS WS profile V11.00.00, section 4.3.2.

Reconfiguration of PIP parameters:

PIP parameters, in particular QoS parameters such as Time-To-Perform, may have to be changed during the lifecycle of PIP implementations. This profile does not make assumptions about whether or not PIP parameters are reconfigurable at run-time.

8. Realization

This section discusses the realization of the requirements defined in sections 6 and 7. Some of these requirements are supported by MMS WS profile as is and therefore are not discussed in detail again. Section 8.1 draws the line between the realization of requirements as defined in MMS and the implementation of additional features as described in this document. The realization of QoS parameters is discussed in section 8.2 as the realization of QoS significantly influences the design of control flow. Section 8.3 then specifies the PIP protocol (cf. section 3.3) using a state machine based notation.

8.1 Execution Contexts and MEPs

The relation between MCC WS profile and MMS WS profile has been discussed in section 4.2. The differing assumptions and requirements of the two profiles cannot freely be combined. In particular, implementing MCC's "strict execution context" for MMS' pure clients is overly complex. Table 1 (page 16) states that pulling-based MEPs are to be implemented using the MMS WS profile. If composition of PIPs is intended then MCC's "strict execution context" shall be assumed. The next sections describe how to implement the MEPs "Synchronous execution" and "Asynchronous execution with callback" (cf. section 7.2) under the "strict execution context". This specification does not provide a separate discussion of realizing these MEPs for the "lax execution context" as the "strict execution context" based implementation fully meets the "lax execution context" requirements. Users may also choose to implement these maps under "lax execution context" using MMS technology.

This specification describes the implementation of non-pulling based MEPs under the "strict execution context". This concerns sections 8.2, 8.3, 10 and 11.

8.2 Realization of QoS (Strict execution context)

This section's contents are aligned with the QoS feature requirements defined in section 6.2. Security and reliable messaging features **MUST** be implemented for each PIP protocol level (cf. section 3.3) message exchanged. These are the messages for exchanging (cf. section 8.3):

- a business document (bizDoc).
- a receipt acknowledgement (RA).
- a receipt acknowledgement exception (RAE).
- a general exception (GE).

Conversely, timing constraints cannot be applied to single message exchanges. Therefore, the realization of "time to acknowledge validity" and "time to perform" as

defined in section 6.2 is described in the next section.

Reliable Messaging

MCC's "strict execution context" requires that integration partners achieve agreement upon the result of PIP executions. The result of PIP executions is determined by the next section's PIP execution protocol that relies on the availability of reliable messaging at the messaging level. Therefore, the use of WS-ReliableMessaging with "Exactly-Once" semantics is compulsory for the exchange of any PIP protocol level message. "At-least-Once" and "At-Most-Once" delivery semantics are considered to be relevant for the "lax execution context" only.

The use of WS-ReliableMessaging with "Exactly-Once" semantics is compulsory for the exchange of all PIP protocol level messages.

Furthermore, it is assumed that a message that is successfully delivered at the messaging level will also successfully be delivered at the application level. Integration partners may apply techniques such as persistent message logs or synchronous delivery for fulfilling this assumption, but the detailed implementation is considered to be out of the scope of this specification.

At runtime, if a WS-ReliableMessaging implementation indicates that the delivery status of the message under transmission is unknown, then the sending party must stop processing until the delivery status has been determined. This specification does not impose restrictions upon how the delivery status may be clarified. Out-Of-Bounds communication is one possible option.

Authentication, Integrity and Confidentiality

The realization of security features is to be performed in a mutual way as well. In particular, if authentication, confidentiality and integrity is requested for a PIP protocol level message *m*, then the following must be true after having exchanged the message (note that reliable messaging always is required):

1. Both the PIP requester and the PIP responder know that *m* has successfully been delivered (Reliable Messaging).
2. *m* has not been modified, i.e., *m* as sent by the PIP requester/responder and *m* as received by the PIP responder/requester are identical. Both PIP requester and PIP responder know that *m* has not been modified (Integrity).
3. *m* has not been disclosed to anybody other than the PIP requester or PIP responder (Confidentiality).
4. The sender of message *m* (PIP requester/responder) is sure about the identity of the message receiver (PIP responder/requester) and, conversely, the receiver of message *m* (PIP responder/requester) is sure about the identity of the message sender (PIP requester/responder).
The message sender/receiver knows that the message receiver/sender has successfully performed an authentication check (Mutual authentication).

Authentication, integrity and confidentiality must be implemented in a mutual way.

The realization of these features is to be realized as pointed out in the WS-ReliableMessaging security considerations (WS-RM sections 5 and 6) using functionality from, among others, WS-Security and WS-SecureConversation and obeying further restrictions from the WS-I Reliable Secure Profile Version 1.0.

For full conformance with this specification, integration partners are free to choose different implementation means as long as the above properties (1-4) hold true. This includes equivalent technologies at the HTTP level. Simple layering of security and reliable messaging features, e.g., using asymmetric signatures as part of the payload of a reliable channel, is not sufficient for the following two reasons:

1. Mutual realization of security-related QoS properties may create a new reliability problem.
Using an asymmetric signature within the payload of a message, the receiver can verify the authenticity of the sender, but the sender cannot be sure about authenticity of the receiver. Sending a signed acknowledgment with a hash value of the original message does not help because then the sender of the acknowledgment cannot be sure who has received the acknowledgment.
2. A malicious attacker must be assumed.
A malicious attacker basically may try to manipulate any message exchanged between integration partners. This not only holds true for the message payloads but for lower-level transport messages as well. This means that an attacker may try to manipulate communication by means of tampering with unsecured reliability messages. While an attacker may not be able to break arbitrary security goals, the reliability property is endangered.

Authorization

Authorization can be split up into system authorization and client/server authorization, where system authorization refers to limiting access to business service interfaces from within the organization and client/server authorization refers to limiting acceptance of PIP protocol level messages from integration partners.

System authorization is out of the scope of this specification.

Client/Server authorization is to be implemented mutually, i.e., after having exchanged a PIP protocol level message *m*, the following must be true:

- The receiver of message *m* (PIP responder/requester) considers *m* to have been sent by an authorized entity.
The sender of message *m* (PIP requester/responder) must be sure that the message receiver (PIP responder/requester) considers *m* to have been sent by an authorized entity.
- The sender of message *m* (PIP requester/responder) considers *m* to have been received by an authorized entity.
The receiver of message *m* (PIP responder/requester) must be sure that the message sender (PIP requester/responder) considers *m* to have been received by an authorized entity.

In the simplest case, authentication is identical with authentication, i.e., any authenticated entity also is authorized. Authorization checks that go beyond authentication have to be integrated with the implementation of authentication such that whenever authentication procedures succeed/fail, then authorization procedures succeed/fail as well.

This is due to similar security considerations as for mutual authentication.

<i>Authorization must be implemented in a mutual way.</i>

Non-Repudiation and Non-Repudiation-Of-Receipt

Non-Repudiation and Non-Repudiation-Of-Receipt are special in implying a very hard error model. Non-repudiation is defined as the property that the sender of a particular message cannot deny having sent the message. The attempt to deny having sent/received a message implies that an integration partner cannot be assumed to behave as defined in a protocol specification. If so, the possibility of implementing two-way non-repudiation is questionable, i.e., the sender cannot deny having sent a message while the receiver cannot deny having received it.

Therefore, non-repudiation is to be implemented in an asymmetric way by attaching a signature to business documents and business signals and archiving these messages upon arrival. Once the non-repudiated message has been archived the receiver can claim to have successfully received the message. If a PIP execution succeeds (which should be the standard case) then having implemented non-repudiation in an asymmetric way does not do any harm. If it fails, the receiver of the non-repudiated message may assert a claim based on the message.

<i>Non-Repudiation/Non-Repudiation-Of-Receipt must be implemented in an asymmetric way.</i>

8.3 Control flow (Strict execution context)

This section specifies the flow of messages for implementing the *Asynchronous execution with callback* MEP as well as the *Synchronous execution* MEP of section 7.2. Further, the realization of all PIP property parameters of table Table 2 that have not yet been covered in the last section is specified. The specification is provided using a *communicating state machine* formalism and the semantics is explained in plain text. Note that all messages **between** PIP requester and PIP responder are exchanged synchronously from a messaging point of view, i.e., there are no message buffers for decoupling the interaction **between** PIP requester and PIP responder (there may be buffers for the internal realization of the PIP level protocol though).

Thus, the distinction between asynchronous and synchronous interaction refers to the coupling between PIP requester process and the PIP responder process and not to the transmission of messages.

In the asynchronous case, the PIP requester delivers the business document to the PIP responder (synchronously) and then waits for the PIP responder to send back any messages. In the meantime, various interactions may take place at the PIP requester's side internally.

In the synchronous case, the PIP requester sends the business document to the PIP

responder and blocks on the communication channel until a response is provided. No internal interactions at the PIP requester's side may take place internally.

8.3.1 Asynchronous Interaction

The interaction between a PIP's requesting role and responding role is specified by the state machines depicted in figure Figure 5 and figure **Figure 6**, respectively. These state machines show the respective participant's process when communicating in a PIP.

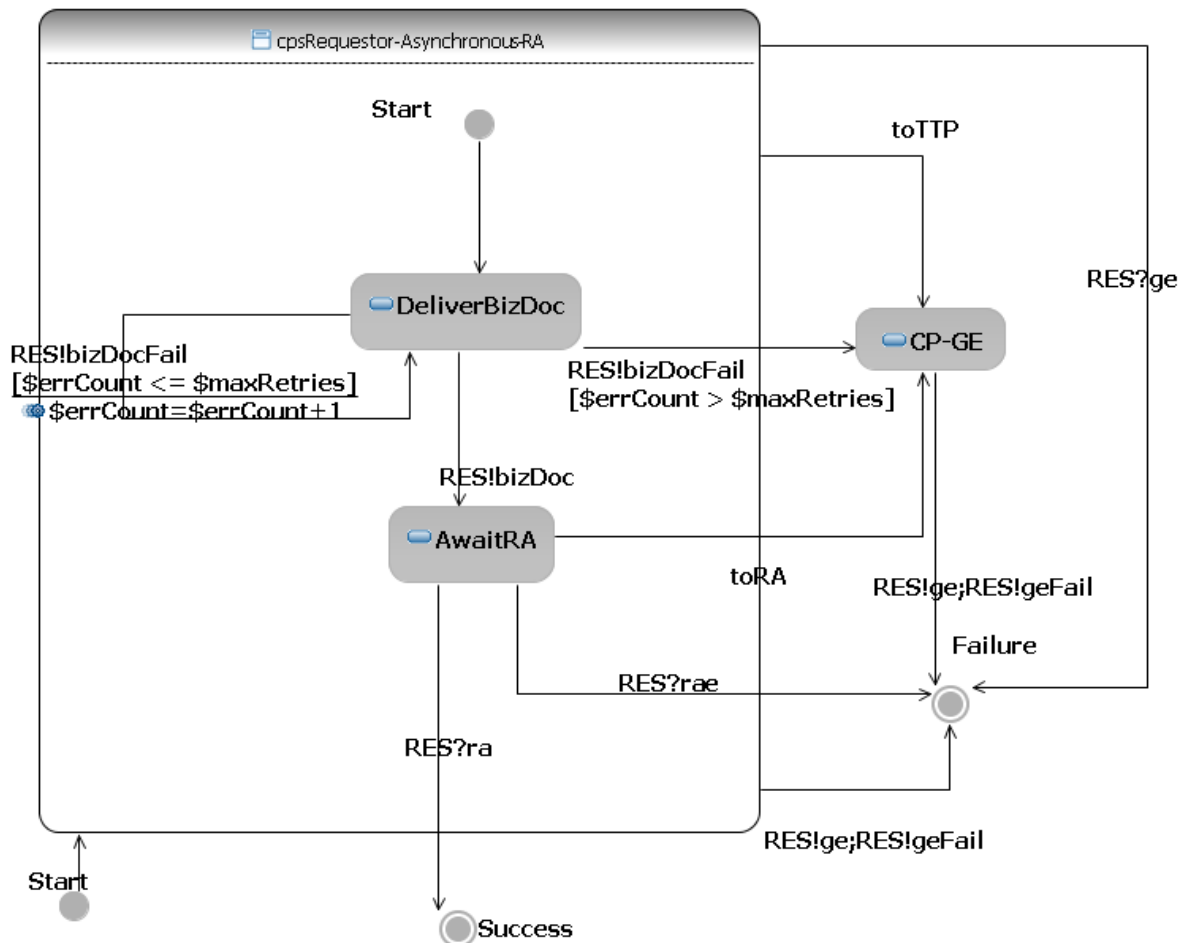


Figure 5: Asynchronous PIP Requester Protocol with RA

PIP Requester Protocol

As shown in figure Figure 5, the requester's process is in state "Deliver Business Document" (DeliverBizDoc) after being started. In this state, the requester attempts to send the business document to the responder's process. If the communication of the business document is successful (RES!bizDoc), the requester's process waits for a receipt acknowledgement from the responder's process (AwaitRA). If the receipt acknowledgement is received within the agreed timeframe (RES?ra), the requester's process finishes successfully.

In the requester's process the following failures may occur:

- *Sending the business document fails (RES!bizDocFail).*
In this case, there are two options:
 - (i) The current business transaction error count is lower than or equals the agreed retry count (RES!bizDocFail[\$errCount <= \$maxRetries]). In this case, the error count is incremented (\$errCount=\$errCount+1) and state "Deliver Business Document" is entered again.
 - (ii) The error count is greater than the agreed retry count (RES!bizDocFail[\$errCount > \$maxRetries]). In this case, the process enters state "General Exception" (CP-GE).
In state "General Exception," the requester's process communicates a general exception to the responder (RES!ge) and terminates with a failure. Sending the general exception may also fail (RES!geFail), but this has no implications on the process (it reaches the state failure).
- *The receipt acknowledgement is not received in time.*
In state "Await Receipt Acknowledgment," the responder waits for the receipt acknowledgment confirming that the responder has properly received the business document. If the time to acknowledge receipt is exceeded (toRA), the requester's process enters state "General Exception" (cf. above).
- *The overall time to perform has exceeded.*
At any time during executing the process, the agreed time to perform may have been exceeded. In this case, the requester's process enters state "General Exception" (cf. above).
- *A receipt acknowledgement exception is received.*
In state "Await Receipt Acknowledgment," the requester's process may receive a receipt acknowledgment exception from the responder (RES?rae). This terminates the requester's process with a failure.
- *A general exception is received.*
As long as a receipt acknowledgement has not been received, the requester may receive a general exception from the responder (RES?ge). This terminates the requester's process with a failure.
- *A general exception is sent.*
As long as a receipt acknowledgement has not been received, the requester may send a general exception to the responder (RES!ge). This corresponds to a cancellation of the process. There may be several reasons for cancelling the process like a cancellation on the business level. Whenever a requester tries to send a general exception, reception of receipt acknowledgements MUST be disabled. In case the responder tries to deliver a receipt acknowledgment at the same time, this call MUST fail. This is possible due to using synchronous communication.
Sending the general exception may also fail (RES!geFail), but this has no implications on the process (it reaches the state failure).

PIP Responder Protocol

After the responder's process has been started, it waits in state "Await Business Document" (AwaitBizDoc) until receiving the business document. If the business document is received in time, the process reaches state "Got Business Document" (GotBizDoc). It is assumed that the business document is safely stored at that point in time. In this state, if the PIP parameterization has the `isIntelligibleCheckRequired` attribute flagged to true (cf. section 7.1), the responder performs the validation steps as agreed upon by the integration partners (cf. *Validity Alignment*, section 6.1.3). Otherwise, no validation procedures are performed.

Assuming that the document has been properly received, the process enters the state "Deliver Receipt Acknowledgement" (DeliverRA). If the receipt acknowledgment is successfully sent to the responder (REQ!ra), the responder's process finishes properly.

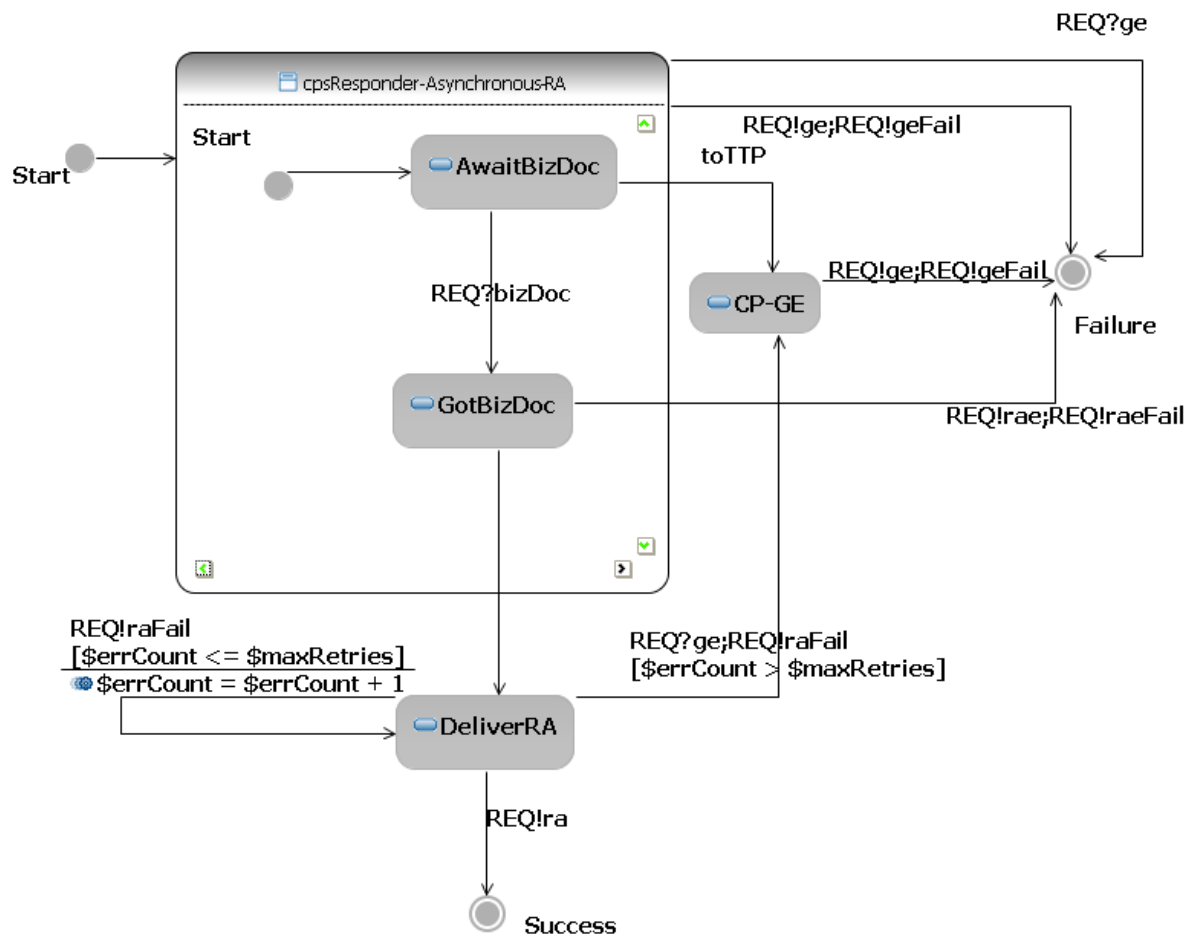


Figure 6: Asynchronous PIP Responder Protocol with RA

In the requester's process the following failures may occur:

- Time to perform may be exceeded.*

In state "Await Business Document," the responder waits for a given timeframe (cf. *Overall Time-To-Perform* as of section 7.1). If the business document is not received within the agreed timeframe (`toTTP`), the state "General Exception" (CP-GE) is entered. In this state, a general exception is sent to the requester (REQ!ge) and the process terminates with a failure.

Sending the general exception may also fail (REQ!raeFail), but this has no implications on the process (it reaches the state failure).

- *The business document checks fail.*
In state "GotBizDoc," the responder checks the received document in case a legibility check is required (cf. *isIntelligibleCheckRequired* of section 7.1). If checking the document fails, a receipt acknowledgement exception is sent to the responder (REQ!rae) and the process terminates unsuccessfully. Sending the receipt acknowledgement exception may also fail (REQ!raeFail), but this has no implications on the process (it reaches the state failure).
- *Delivering the receipt acknowledgement fails.*
In state "Deliver Receipt Acknowledgement" (DeliverRA), the responder attempts to send the receipt acknowledgement to the requester. There are two options if sending the receipt acknowledgement is not successful (REQ!raFail):
 - (i) The internal error count is lower than or equals the agreed retry count (REQ!raFail\$errCount <= \$maxRetries]). In this case, the internal error count is incremented and state "Deliver Receipt Acknowledgement" is entered again.
 - (ii) The internal error count is greater than the retry count (\$errCount > \$maxRetries). In this case, the state "General Exception" (CP-GE, cf. above) is entered.
- *A general exception is received.*
At any time during the execution of the process, the responder may receive a general exception from the requester (REQ?ge). This terminates the responder's process with a failure.
- *A general exception is sent.*
At any time during the execution of the process (except for state DeliverRA), the responder may send a general exception to the requester (REQ!ge). This corresponds to a cancellation of the process. There may be several reasons for cancelling the process like a cancellation on the business level. Sending the general exception may also fail (REQ!geFail), but this has no implications on the process (it reaches the state failure).

Protocol Specifications without ReceiptAcknowledgement

PIP implementers may choose not to use ReceiptAcknowledgement messages for performing PIPs (cf. table Table 2). In this case the state machines depicted in figures Figure 7 and Figure 8 specify the control flow of the PIP requester and the PIP responder process. The semantics of the state machine transitions corresponds to the descriptions above, but the realization of *IntelligibleCheckRequired*, *TimeToAcknowledgeReceipt* and *Non-Repudiation-of-Receipt* does not apply due to the missing ReceiptAcknowledgement (cf. table Table 2).

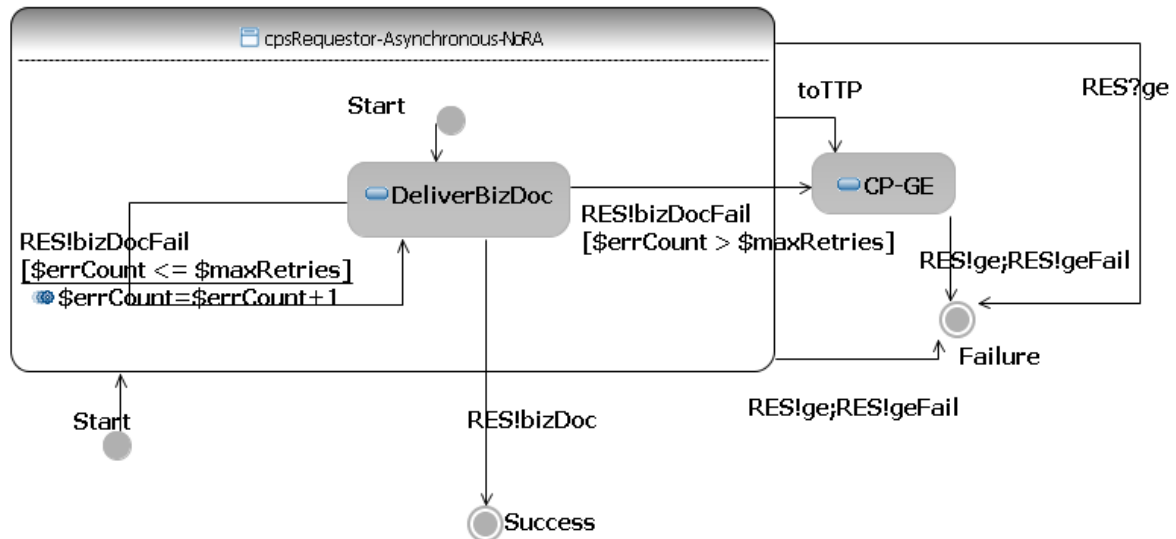


Figure 7: Asynchronous PIP Requester Protocol without RA

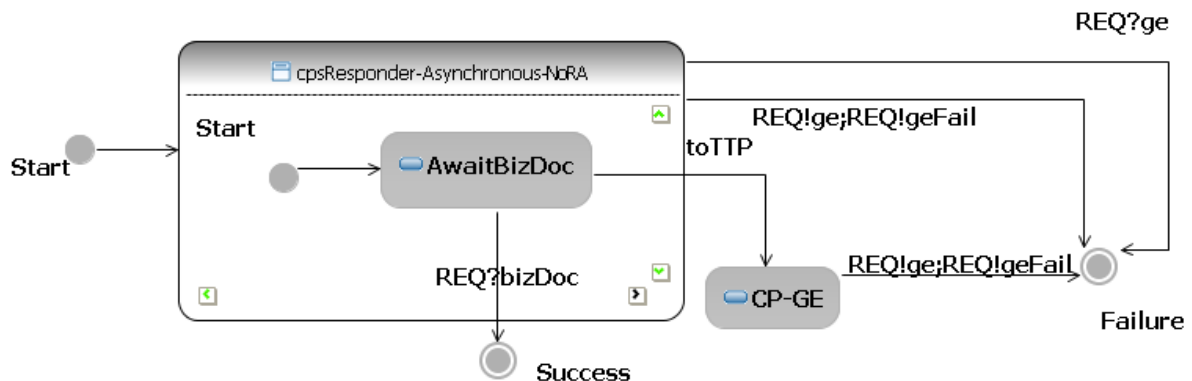


Figure 8: Asynchronous PIP Responder Protocol without RA

8.3.2 Synchronous Interaction

The control flow for implementing synchronous interaction PIPs without ReceiptAcknowledgement corresponds to the control flow for implementing asynchronous interaction PIPs without ReceiptAcknowledgement (cf. figures Figure 7 and Figure 8).

In synchronous interaction PIPs with ReceiptAcknowledgement, acknowledgments and acknowledgment exceptions are communicated synchronously as depicted in figures Figure 9 and Figure 10. Note that the transitions to the Success states in figures Figure 9 and Figure 10 denote the only difference to the protocol specifications for asynchronous interaction PIPs without ReceiptAcknowledgement (cf. figures Figure 7 and Figure 8). This means that sending a business document

opens a synchronous channel and the corresponding acknowledgment (or acknowledgement exception) is sent as response over the same channel.

This is different to asynchronous interaction PIPs, where business documents and acknowledgments are communicated via different communication channels.

In case the *IntelligibleCheckRequired* (cf. section 7.1) parameter has been flagged to true, PIP responder process must perform the validation steps agreed upon by the implementation partners (cf. section 6.1.3) while processing the PIP requester's business document send call. Processing of the *TimeToAcknowledgeReceipt* (cf. section 7.1) parameter is constrained by the timeout handling capabilities of the underlying communication channel.

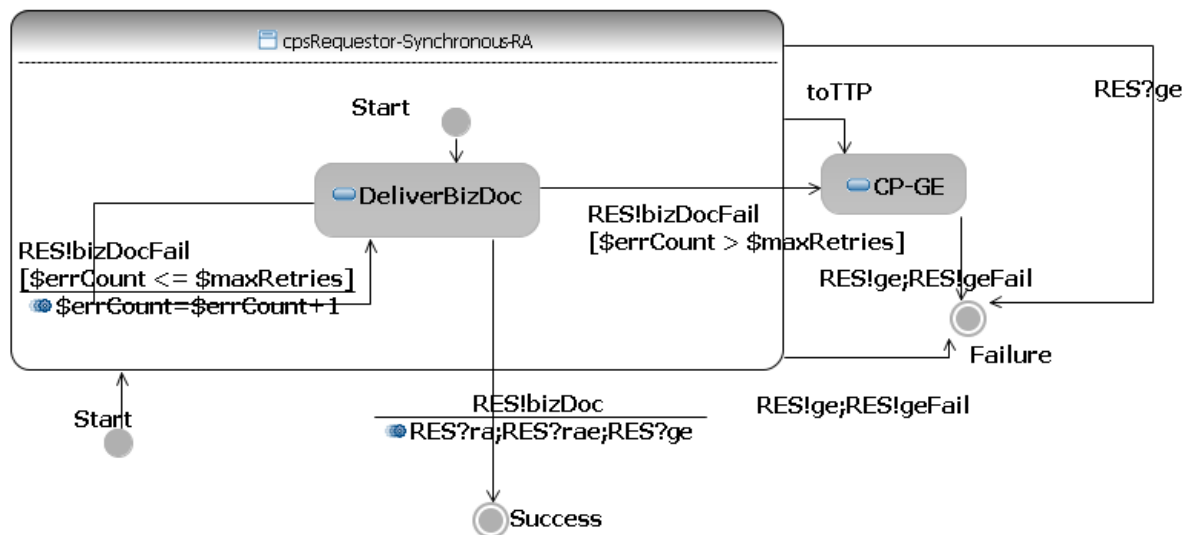


Figure 9: Synchronous PIP Requester Protocol with RA

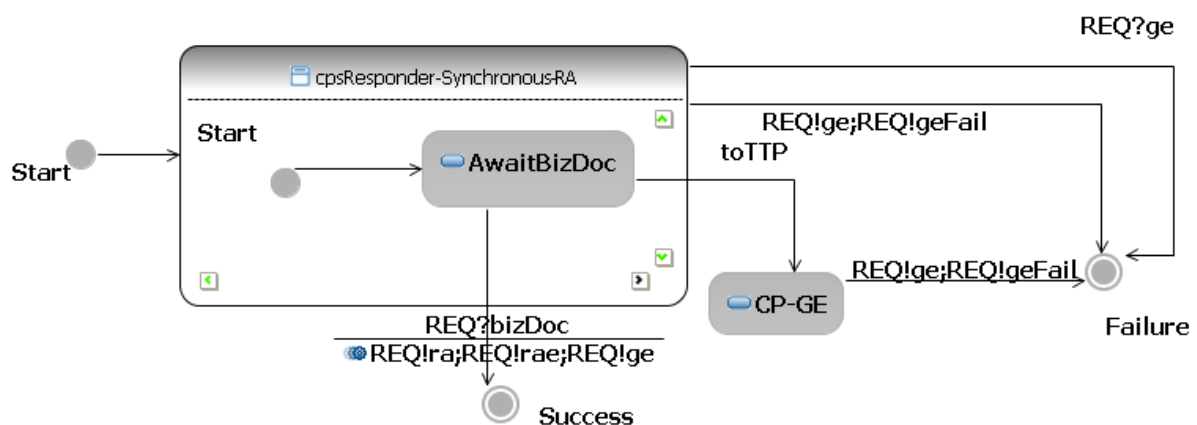


Figure 10: Synchronous PIP Responder Protocol with RA

8.3.3 PIP Instance Identification and Message Correlation

Message correlation denotes the act of associating messages with process instances which may be implemented at the SOAP/HTTP level or at the PIP process level. While message correlation at the SOAP/HTTP level leverages the intrinsic correlation features of the messaging technology, message correlation at the PIP process level defines message correlation in terms of the content of PIP business document headers and business signal headers, which offers more flexibility with respect to usage of a particular messaging technology.

In this section, the following abbreviations will be used:

bizDoc	=	Business Document
RA	=	ReceiptAcknowledgement
RAE	=	ReceiptAcknowledgmentException
GE	=	GeneralException

Correlation Requirements

The choice of correlation mechanism depends on the style of interaction. As the MCC effort targets at the composition of multiple PIPs within a single collaboration the following situations have to be accommodated for (considering the interaction styles of sections 8.3.1 and 8.3.2):

- Group I: Isolated PIPs
Simple B2B interactions may consist of one single PIP (isolated PIPs). Then, the following requirements for correlating PIP level messages may arise:
 - Case I1: Synchronous interaction without RA
In this case, the PIP requester tries to send the PIP bizDoc to the PIP responder and immediately terminates when the message transmission is terminated.
As there is no superordinate process instance, the PIP responder process effectively starts and terminates by processing the Web service call carrying the bizDoc.
The transmission of GEs as defined in section 8.3.2 therefore does not apply.
 - Case I2: Asynchronous interaction without RA
The correlation requirements of case I2 are identical to the requirements of case I1.
 - Case I3: Synchronous interaction with RA
In this case, the PIP requester tries to send the PIP bizDoc to the PIP responder and blocks on the corresponding Web service call until a RA, a RAE or GE is provided by the PIP responder. Thus, business document and reply message (RA, RAE or GE) have to be correlated.
As there is no superordinate process instance, the PIP responder process effectively starts and terminates by processing the Web service call carrying the bizDoc.
The transmission of GEs outside of the bizDoc Web service call (as defined in section 8.3.2) therefore does not apply.

- Case I4: Asynchronous interaction with RA
In this case, the PIP requester tries to send the PIP bizDoc to the PIP responder in a first Web service call and, if successful, waits for an incoming Web service call for collecting a RA, RAE or GE message. As there is no superordinate process instance, the PIP responder process effectively starts upon reception of the bizDoc and terminates upon either delivering a RA, RAE or GE to the PIP requester or by receiving a GE from the PIP requester (cf. figure **Figure 6**). Therefore, correlation between bizDoc and reply message (RA, RAE, GE) in the PIP requester's process as well as correlation between bizDoc and (potentially) an additional GE at the PIP responder's process has to be performed.
- Group C: Composable PIPs
Complex B2B interactions may be composed of several PIPs (denoted as *business collaborations*). In this case, PIP level messages not only have to be correlated with each other, but also with the superordinate business collaboration. Moreover, coordination between business collaboration processes and subordinate PIP processes may require starting PIP responder processes before the bizDoc has been transmitted. Decoupling the start of PIP requester and PIP responder processes from the transmission of bizDoc messages is explicitly supported by the PIP protocol defined in sections 8.3.1 and 8.3.2. In turn, GE messages that are transmitted before the bizDoc message may have to be correlated to a running PIP instance as well as a superordinate business collaboration instance.
 - Case C1: Synchronous interaction without RA
All messages defined in figures Figure 7: Asynchronous PIP Requester Protocol without RA and Figure 8 (these figures apply to synchronous interaction as well) have to be correlated with the running PIP instance.
 - Case C2: Asynchronous interaction without RA
All messages defined in figures Figure 7: Asynchronous PIP Requester Protocol without RA and Figure 8 have to be correlated with the running PIP instance.
 - Case C3: Synchronous interaction with RA
All messages defined in figures Figure 9 and Figure 10 have to be correlated with the running PIP instance.
 - Case C4: Asynchronous interaction with RA
All message defined in figures Figure 5 and Figure **6** have to be correlated with the running PIP instance.

The following explanations specify the choice of correlation mechanism for the above scenarios of PIP execution. Note that, the definition of message correlation with business applications/backend systems is out of the scope of this specification.

Moreover, the assumption of the "*MCC Phase 1 – Single Business Document PIP Template*" document about generation of PIP instance identifiers is required:

"Generation of Globally Unique Ids (GUIDs) for PIP instances

PIP instance ids are to be generated by the PIP requester by appending an id that is unique within her systems to her globally unique partner id, preferably a GLN or a DUNS number."

Realization of Message Correlation

In case I1 and case I2, the bizDoc message essentially is the only message to be exchanged. Therefore, there is no need for a correlation mechanism.

In case I3, all messages (bizDoc, RA, RAE and GE) are transmitted within a single request-reply Web service call. Therefore, the correlation between bizDoc and RA, RAE or GE is inherently provided by the messaging technology. For flexibility reasons, this specification encourages the use of the correlation mechanism defined for case I4.

In case I4, the bizDoc message initiates the overall process. This specification defines content based correlation between messages (bizDoc, RA, RAE, GE) in terms of the document identifier included in the bizDoc message. This identifier **MUST** be a GUID as defined above and is to be included in the messages exchanged as follows:

- For DTD-based PIP bizDoc messages:
<ROOT-TAG-OF-PIP>.<thisDocumentIdentifier>.<ProprietaryDocumentIdentifier>
- For XSD-based PIP bizDoc messages:
<ROOT-TAG-OF-PIP>.<DocumentHeader>.<DocumentInformation>.<DocumentIdentification>.<Identifier>
- For RA messages (cf. ebBP ReceiptAcknowledgement definition):
<ReceiptAcknowledgement>.<OriginalDocumentIdentifier>
- For RAE messages (cf. ebBP *Exception* definition):
<Exception>.<OriginalDocumentIdentifier>
- For GE messages (cf. ebBP *Exception* definition):
<Exception>.<OriginalDocumentIdentifier>

As an alternative to content based message correlation, partners may agree to use WS-Addressing as defined in MMS (cf. MMS section 3.5 *Message Correlation*), which corresponds to defining relations between PIP level messages at the transport level. Consequently, the use of WS-Addressing is discouraged by this specification.

Full conformance with this specification requires support for content-based message correlation for case I4.

In cases C1, C2, C3 and C4, this specification defines content based correlation between messages (bizDoc, RA, RAE, GE) in terms of an additional *composition* header that is to be contained together with the actual PIP payload message within a *composition* container. The *composition header* is to be created by a superordinate process instance that controls the overall business collaboration. Detailed rules for creating and processing the *composition header* therefore are to be defined by MCC phase 2. For MCC phase 1, correlation of PIP level messages in terms of the composition header is relevant only.

The *composition header* is defined as follows:

```
<xsd:complexType name="commonCompositionHeaderType">
  <xsd:sequence>
    <xsd:element name="RootIdentifier"
      type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="ParentIdentifier"
      type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="InstanceIdentifier"
      type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="ProcessDepth"
      type="xsd:int" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

For PIP level message correlation, the **InstanceIdentifier** field is to be used. Further, MCC phase 1 implementations must not make any assumptions about content or processing of all composition header fields except for the **InstanceIdentifier** field. In particular, any other message correlation mechanism, including WS-Addressing, is disallowed (cases C1, C2, C3, C4).

For cases C1, C2, C3 and C4, this specification mandates:

- the use of the composition header's **InstanceIdentifier** field for PIP level message correlation.*
- WS-Addressing **MUST NOT** be used for PIP level message correlation.*

9. WSDL Mapping Rules

Web Service interfaces defined using WSDL provide information on the endpoints & the messages in an implementation-independent fashion; i.e. the data being exchanged (messages) & the operations exposed are treated as an abstract description independent of the message format or the network protocol used.

As defined in section 3.1, an abstract WSDL file has type definitions, message definitions & port type definitions. These definitions, when extended by providing binding information on concrete protocol, network addresses & port information for usage of the web service, form the concrete WSDL definition. This section will define the abstract part of a WSDL definition with references to the binding information present in the MMS WS Profile specification.

MMS WS profile specification uses WSDL specification version 1.1 & defines the IT scenarios or the Message Exchange Patterns (MEP) in its sections 3.2 and 3.3.

This specification adopts the WSDL mapping rules of the MEPs defined in section 3.3 of the MMS WS profile in full (this also covers the **Asynchronous execution with pulling** MEP of section 7.2). For the **Service-to-Service** MEP of the MMS WS profile, slight adaptations have to be made for accommodating both the **Synchronous Execution** and **Asynchronous execution with callback** of section 7.2, and for accommodating the message correlation requirements of section 8.3.3.

In the following sections, the adaptations to the MMS Service-to-Service MEP will be provided by summarizing the relevant WSDL mapping rules as defined in MMS and by defining new MCC phase 1 rules. For clarity, MMS rules will be indexed by ids of the form **RXXXX** (directly taken from the MMS profile) and MCC phase 1 rules will be indexed by ids of the form **MCCRXXXX**.

9.1 Messages

9.1.1 Importing Message Types

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

For clarity, this requirement applies to the following objects of specification:

- XSD based PIPs.
- DTD based PIPs.
For the purpose of importing DTD PIP type definitions, conversion XSDs provided by RosettaNet must be used (cf. section 3.7).
- Composition header extensions.
In case composition headers as defined in section 8.3.3 are to be used, XSD PIP or DTD PIP type definitions have to be wrapped within a composition container.

When adding a composition header to an existing RosettaNet business document definition, a new XML Schema element is to be created and its name has to be built from prepending the prefix "**Composable**" in front of the original RosettaNet type definition. The new *composable* XML Schema element must contain the composition header and the original PIP type definition of the business document's root element. An example for a composable type definition is given for PIP 3A20:

```
<xs:element
  name="ComposablePip3A20PurchaseOrderConfirmationNotification">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="head:TransactionCompositionHeader"/>
      <xs:element
        ref="pip:Pip3A20PurchaseOrderConfirmationNotification"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The new composable XML Schema element must be defined within an XML namespace that is derived from the original namespace by appending the string **":composable"**.

MCCR1001 Signal message types, i.e. receipt acknowledgement, receipt acknowledgement exception and general exception, have to be imported from the ebBP specification.

MCCR1002 In case a composable PIP is to be defined, the predefined RosettaNet composition containers that carry ebBP signal definitions, have to be used.

9.1.2 Defining WSDL Messages

The following rules apply to the WSDL messages that refer to the business document schemas & the signal schemas defined by RosettaNet.

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

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

MCCR1003 The single part MUST refer to the composition container element in case a composable execution has to be assumed.

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.

Examples for defining WSDL messages for Isolated PIPs or Composable PIPs (cf. section 8.3.3) are given below:

Example:

```
xmlns:ns="urn:rosettanet:specification:interchange:PurchaseOrderConfirmationNotification:dtdbase:01.00"
```

...

```
<wsdl:message  
  name="Pip3A20PurchaseOrderConfirmationNotificationMsg">  
  <wsdl:part name="Pip3A20PurchaseOrderConfirmationNotificationPart"  
    element="ns:Pip3A20PurchaseOrderConfirmationNotification"/>  
</wsdl:message>
```

Example:

```
xmlns:ns="urn:rosettanet:specification:interchange:PurchaseOrderConfirmationNotification:dtdbase:01.00:composable"
```

...

```
<wsdl:message  
  name="ComposablePip3A20PurchaseOrderConfirmationNotificationMsg">  
  <wsdl:part  
    name="ComposablePip3A20PurchaseOrderConfirmationNotificationPart"  
    element="ns:ComposablePip3A20PurchaseOrderConfirmationNotification"  
  />  
</wsdl:message>
```

The definition of PIP signal messages as WSDL messages is analogous to the definition of business messages. The message definition preserves the generic definition of exceptions as defined in ebBP and therefore does not distinguish between receipt acknowledgement exception messages and general exceptions.

<i>MCCR1004 For Isolated PIPs, the raw ebBP signal definitions have to be used.</i>

Example:

```
xmlns:sig="http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0"
```

...

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

<wsdl:message name="ReceiptAcknowledgementMsg">
  <wsdl:part name="ReceiptAcknowledgementPart"
    element="sig:ReceiptAcknowledgement"/>
</wsdl:message>
```

MCCR1005 For Composable PIPs, the ebBP signal definitions within a RosettaNet composition container have to be used.

Example:

```
xmlns:ctrl="urn:rosettanet:specification:interchange:composable:xml:controlMsg:1.0"

<wsdl:message name="ComposableReceiptAcknowledgementMsg">
  <wsdl:part name="ComposableReceiptAcknowledgementPart"
    element="ctrl:ComposableReceiptAcknowledgementMessage"/>
</wsdl:message>

<wsdl:message name="ComposableExceptionMsg">
  <wsdl:part name="ComposableExceptionPart"
    element="ctrl:ComposableExceptionMessage"/>
</wsdl:message>
```

9.2 Operations

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***

MCCR1006 For operations with an input and output RosettaNet Business Message the operation name MUST be constructed by appending 'Op' to the root element of the input RosettaNet schema. The convention used is the same as for rule R1009.

MCCR1007: Operation names for composable PIPs MUST be constructed by adding the prefix 'Composable' and the suffix 'Op' to the root element of the RosettaNet/ebBP type definition.

*MCCR1008: For synchronous interactions, the PIP responder's receipt acknowledgment exception or general exception in reply to the PIP requester's business message MUST be defined as WSDL **fault** detail.*

Example:

```
<wsdl:operation
  name="Pip3A20PurchaseOrderConfirmationNotificationOp">
  <wsdl:input name="input"
    message="tns:Pip3A20PurchaseOrderConfirmationNotificationMsg"/>
</wsdl:operation>
```

Example:

```
<wsdl:operation
  name="ComposablePip3A20PurchaseOrderConfirmationNotificationOp">
  <wsdl:input name="input"
    message="tns:ComposablePip3A20PurchaseOrderConfirmationNotification
Msg"/>
</wsdl:operation>
```

Example:

```
<wsdl:operation
  name="ComposablePip3A20PurchaseOrderConfirmationNotificationOp">
  <wsdl:input name="input"
    message="tns:ComposablePip3A20PurchaseOrderConfirmationNotificationMs
g"/>
  <wsdl:output name="output"
    message="tns:ComposableReceiptAcknowledgementMsg"/>
  <wsdl:fault name="fault" message="tns:ComposableExceptionMsg"/>
</wsdl:operation>
```

Example:

```
<wsdl:operation
  name="Pip3A20PurchaseOrderConfirmationNotificationOp">
  <wsdl:input name="input1"
    message="tns:Pip3A20PurchaseOrderConfirmationNotificationMsg"/>
  <wsdl:output name="output1"
    message="tns:ReceiptAcknowledgementMsg"/>
  <wsdl:fault name="fault1" message="tns:ExceptionMsg"/>
</wsdl:operation>
```

*R1011 Fault operation MUST be named '**ExceptionOp**'*

*R1012 Receipt Acknowledgement operation MUST be named '**ReceiptAcknowledgmentOp**'*

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

R1014 ExceptionOp MUST have only 'ExceptionMsg' as the input.

Example:

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

Example:

```
<operation name="ReceiptAcknowledgementOp">  
  <input name="input" message="tns:ReceiptAcknowledgementMsg"/>  
</operation>
```

Example:

```
<operation name="ComposableExceptionOp">  
  <input name="input" message="tns:ComposableExceptionMsg"/>  
</operation>
```

Example:

```
<operation name="ComposableReceiptAcknowledgementOp">  
  <input name="input"  
message="tns:ComposableReceiptAcknowledgementMsg"/>  
</operation>
```

9.2.1 Operations Required for Mapping Message Exchange Patterns

The WSDL portType definitions for different MEPs and message correlation requirements are given below. Table 3 serves for looking up the correct portType definition given the MEP as of section 7.2 and correlation requirement as of section 8.3.3. Note that MEPs not listed here are to be performed as defined in the MMS WS profile. Note further that portType names are not subject to standardization.

MEP	Correlation Requirement	Uses ReceiptAck.	PIP requester	PIP responder
Asynchronous execution with callback	Isolated	Yes	Definition 1	Definition 2
Asynchronous execution with callback	Composable	Yes	Definition 3	Definition 4
Synchronous execution	Isolated	Yes	Definition 5	Definition 6
Synchronous execution	Composable	Yes	Definition 7	Definition 8
Asynchronous execution (callback not applicable)	Isolated	No	Definition 9	Definition 10
Asynchronous execution (callback not applicable)	Composable	No	Definition 11	Definition 12
Synchronous execution	Isolated	Yes	Definition 13	Definition 14
Synchronous execution	Composable	Yes	Definition 15	Definition 16

Table 3: PortType lookup table

Definition 1 (Asynch. Execution/Isolated/RA/PIP requester):

```
<wsdl:portType name="PIP3A20RAAsynchRequestorPortType">
  <wsdl:operation name="ReceiptAcknowledgementOp">
    <wsdl:input name="input1"
message="tns:ReceiptAcknowledgementMsg"/>
  </wsdl:operation>
  <wsdl:operation name="ExceptionOp">
    <wsdl:input name="input2" message="tns:ExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 2 (Asynch. Execution/Isolated/RA/PIP responder):

```
<wsdl:portType name="PIP3A20RAAsynchResponderPortType">
  <wsdl:operation
name="Pip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:Pip3A20PurchaseOrderConfirmationNotificationMsg"/>
  </wsdl:operation>
  <wsdl:operation name="ExceptionOp">
    <wsdl:input name="input2" message="tns:ExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 3 (Asynch. Execution/Composable/RA/PIP requester):

```
<wsdl:portType name="PIP3A20RAAsynchRequestorPortType">
  <wsdl:operation name="ComposableReceiptAcknowledgementOp">
    <wsdl:input name="input1"
message="tns:ComposableReceiptAcknowledgementMsg"/>
  </wsdl:operation>
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input2"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 4 (Asynch. Execution/Composable/RA/PIP responder):

```
<wsdl:portType name="PIP3A20RAAsynchResponderPortType">
  <wsdl:operation
name="ComposablePip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:ComposablePip3A20PurchaseOrderConfirmationNotificationMs
g"/>
  </wsdl:operation>
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input2"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 5 (Synch. Execution/Isolated/RA/PIP requester):

NOT DEFINED

Definition 6 (Synch. Execution/Isolated/RA/PIP responder):

```
<wsdl:portType name="PIP3A20RASynchResponderPortType">
  <wsdl:operation
name="Pip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:Pip3A20PurchaseOrderConfirmationNotificationMsg"/>
    <wsdl:output name="output1"
message="tns:ReceiptAcknowledgementMsg"/>
    <wsdl:fault name="fault1" message="tns:ExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 7 (Synch. Execution/Composable/RA/PIP requester):

```
<wsdl:portType name="PIP3A20RASynchRequestorPortType">
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input1"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 8 (Synch. Execution/Composable/RA/PIP responder):

```
<wsdl:portType name="PIP3A20RASynchResponderPortType">
  <wsdl:operation
name="ComposablePip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:ComposablePip3A20PurchaseOrderConfirmationNotificationMs
g"/>
    <wsdl:output name="output1"
message="tns:ComposableReceiptAcknowledgementMsg"/>
    <wsdl:fault name="fault1"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input2"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 9 (Asynch. Execution/Isolated/No RA/PIP requester):

NOT DEFINED

Definition 10 (Asynch. Execution/Isolated/No RA/PIP responder):

```
<wsdl:portType name="PIP3A20AsynchResponderPortType">
  <wsdl:operation
name="Pip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:Pip3A20PurchaseOrderConfirmationNotificationMsg"/>
  </wsdl:operation>
</wsdl:portType>
```


Definition 11 (Asynch. Execution/Composable/No RA/PIP requester):

```
<wsdl:portType name="PIP3A20AsynchRequestorPortType">
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input1"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 12 (Asynch. Execution/Composable/No RA/PIP responder):

```
<wsdl:portType name="PIP3A20AsynchResponderPortType">
  <wsdl:operation
name="ComposablePip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:ComposablePip3A20PurchaseOrderConfirmationNotificationMs
g"/>
  </wsdl:operation>
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input2"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 13 (Synch. Execution/Isolated/No RA/PIP requester):

NOT DEFINED

Definition 14 (Synch. Execution/Isolated/No RA/PIP responder):

```
<wsdl:portType name="PIP3A20SynchResponderPortType">
  <wsdl:operation
name="Pip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:Pip3A20PurchaseOrderConfirmationNotificationMsg"/>
    <wsdl:fault name="fault1" message="tns:ExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 15 (Synch. Execution/Composable/No RA/PIP requester):

```
<wsdl:portType name="PIP3A20SynchRequestorPortType">
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input1"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

Definition 16 (Synch. Execution/Composable/No RA/PIP responder):

```
<wsdl:portType name="PIP3A20SynchResponderPortType">
  <wsdl:operation
name="ComposablePip3A20PurchaseOrderConfirmationNotificationOp">
    <wsdl:input name="input1"
message="tns:ComposablePip3A20PurchaseOrderConfirmationNotificationMs
g"/>
    <wsdl:fault name="fault1"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
  <wsdl:operation name="ComposableExceptionOp">
    <wsdl:input name="input2"
message="tns:ComposableExceptionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

10. Use Cases of PIP Definition (Strict execution context)

This section shows two sample configurations of PIPs according to the configurability matrix in section 7.1. Both of the use cases in this section are examples of the Strict Execution Model, which requires that integration partners achieve agreement upon the result of PIP executions. Both use cases involve usage of several of the Quality of Service properties discussed in this document.

10.1 Use Case 1 – Full features

Use Case 1 shows a sample configuration for a Purchase Order Confirmation PIP, with full Quality of Service features included, along with a Receipt Acknowledgement. A verbal description follows below the configuration for Use Case 1 shown here:

```
<DataExchange
  name="bt-PIP3A20"
  nameID="bt-PIP3A20"
  isGuaranteedDeliveryRequired="true">
    <RequestingRole name="Purchase Order Confirmation Sender"
nameID="bt-PIP3A20-role-sender"/>
    <RespondingRole name="Purchase Order Confirmation Receiver"
nameID="bt-PIP3A20-role-receiver"/>
    <RequestingBusinessActivity
      name="Send Purchase Order Confirmation"
      nameID="bt-PIP3A20-ba-req"
      isIntelligibleCheckRequired="true"
      isNonRepudiationRequired="true"
      isNonRepudiationReceiptRequired="true"
      retryCount="3"
      timeToAcknowledgeReceipt="PT3M"
      isAuthorizationRequired="true"
    >
      <DocumentEnvelope
        name="doc-PIP3A20-PurchaseOrderConfirmation"
        businessDocumentRef="doc-PIP3A20-
PurchaseOrderConfirmation"
        nameID="doc-PIP3A20-PurchaseOrderConfirmation-de"
        isAuthenticated="transient"
        isConfidential="transient"
        isTamperDetectable="transient"
      />
      <ReceiptAcknowledgement
        name="ra"
        nameID="bt-PIP3A20-ack-ra"
        signalDefinitionRef="ra2"/>
      <ReceiptAcknowledgementException
        name="rae"
        nameID="bt-PIP3A20-ack-rae"
        signalDefinitionRef="rae2"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name="xsd-pacifier" nameID="bt-
```

```
PIP3A20-ba-resp" />  
</DataExchange>
```

This use case utilizes many of the Quality of Service configurations available, and also involves a Receipt Acknowledgement for business state alignment. These configurations are described below, with some definitions coming directly from the ebXML Business Process Specification Schema Technical Specification. Please also see [section 8.2: Realization of QoS](#) above for further information on realization of these Quality of Service features with the WS MCC Profile.

Receipt Acknowledgement for Business State Alignment:

- A ReceiptAcknowledgement requirement is defined in the configuration for Use Case 1 like so: <ReceiptAcknowledgement name="ra" nameID="bt-PIP3A20-ack-ra" signalDefinitionRef="ra2"/>
The Receipt Acknowledgment is used to confirm that the Purchase Order Confirmation message has been successfully received at the PIP protocol layer at the receiver side.
- The isIntelligibleCheckRequired attribute is set to "true" for the RequestingBusinessActivity.
This means that the partners participating in this business transaction are agreeing that a Receipt Acknowledgment should confirm a message only if it is also legible, i.e. that it has passed validity checks as defined in section 6.1.3.
- The ReceiptAcknowledgementException is defined like so:
<ReceiptAcknowledgementException name="rae" nameID="bt-PIP3A20-ack-rae" signalDefinitionRef="rae2"/>
This means that the partners participating in this business transaction are agreeing to send a ReceiptAcknowledgementException in case the incoming PIP business document cannot be processed correctly. In particular, as the isIntelligibleCheckRequired attribute is set to "true", partners have to send a ReceiptAcknowledgementException in case the validity checks as defined in section 6.1.3 cannot be performed successfully.

Security Features:

- isAuthorizationRequired is set to "true" for RequestingBusinessActivity.
This specifies that the PO Confirmation message must only be processed as valid if the receiving party successfully matches the stated role of the activity to a list of allowed values previously supplied by the requesting party. In this use case, the Requesting Role name is "Purchase Order Confirmation Sender" and the Requesting Role ID is "bt-PIP3A20-role-sender." Authorization is described in further detail in [section 8.2](#) above.
- Document Envelope settings:
 - isAuthenticated is set to "transient."
This means that authentication is implemented at the communication (SOAP/HTTP) level. The specific method is determined by the communications protocol used. If this had been set to "persistent," it

would mean that *"the Business Document signer's identity must be verified at the receiving application level"* (cf. ebBP spec.), to assist in verification of role identity.

Note that the configuration option "persistent" is NOT allowed for in the Strict Execution Model.

- isConfidential is also set to "transient."
Transient confidentiality is provided by a secure network protocol, as the message is transferred between two adjacent transport messaging nodes. In comparison, "persistent" confidentiality would mean that *"the message must remain in encrypted form after it is delivered to the messaging node and will be decrypted only by the authorized application"* (cf. ebBP spec.).
Note that the configuration option "persistent" is NOT allowed for in the Strict Execution Model.
- isTamperDetectable is set to "transient."
This provides the ability to detect if the information has been tampered with during transfer between two adjacent transport messaging nodes.
If set to "persistent," it would provide the *"ability to detect if the information has been tampered with after it has been received by messaging node, between the messaging node and the application. Tamper detection assists in verification of content integrity between participating parties and within a participating party"* (cf. ebBP spec.).
Note that the configuration option "persistent" is NOT allowed for in the Strict Execution Model.

Reliable Messaging:

- isGuaranteedDeliveryRequired is set to "true" for the Data Exchange.
This is a declaration that the sender and receiver must employ only a delivery channel that provides a delivery guarantee for this business transaction's business messages AND business signals.

Non Repudiation and Non Repudiation of Receipt:

- isNonRepudiationRequired is set to "true" for the RequestingBusinessActivity.
This means that the PIP requester must add a fresh digital signature to the business document message so that she cannot claim later not to have sent the message or to have sent a message with different content. The PIP responder, in turn, is assumed to persistently store the business document message together with its signature.
- IsNonRepudiationReceiptRequired is set to "true" for the RequestingBusinessActivity.
This setting requires the receiver of the PO Confirmation document to send a signed receipt, which the original PIP requester saves. In order for the NonRepudiationOfReceipt mechanism to function properly, ReceiptAcknowledgement must also be required in the business transaction (which it is in this use case). This is because it requires the Receipt Ack to be

digitally signed or a comparable mechanism be used.

"If a timeToAcknowledgeReceipt is also imposed on a requesting message (which it also is here), only a digitally signed (or comparable mechanism) receipt will satisfy the imposed timeout deadline. Thus, a Failure to send a signed receipt within timeToAcknowledgeReceipt, would make the transaction null and void, i.e. the agreed upon expectations of business significance of the Requesting party has not been adhered to in the activity" (cf. ebBP spec.).

Timing Constraints:

- timeToAcknowledgeReceipt is set to "PT3M" for the RequestingBusinessActivity.
"PT3M" stands for "Period of Time of 3 Minutes." This means that the Receiver of the PO Confirmation message must send a Receipt Acknowledgement within 3 minutes. The start point is the time the PO Confirmation document is sent by the Requesting Party, and the end point is the time until the time that the Receipt Acknowledgement is properly received by the Requesting party.

retryCount is set to "3" for Use Case 1, which means that the transmission of this message from the sender will be attempted three times before the PIP is considered as failed.

Based on the above descriptions, it can be seen that Use Case 1 is an example of the full usage of the Quality of Service features. In addition, as is described in section 8.2, it is assumed that a message that is successfully delivered at the messaging level will also successfully be delivered at the application level. In Use Case 1 we are making use of the Receipt Acknowledgement (without any specification for an Acceptance Acknowledgement, which is not covered by the MCC WS Profile document).

10.2 Use Case 2 – Business Document Only

Use Case 2 shows a sample configuration for a Purchase Order Confirmation PIP, which also follows the Strict Execution model. However, in comparison with Use Case 1, it implements less of the Quality of Service and Business State Alignment features, and it also does not specify a requirement for a Receipt Acknowledgement. A verbal description and comparison with Use Case 1 follows below:

```
<DataExchange
  name="bt-PIP3A20"
  nameID="bt-PIP3A20"
  isGuaranteedDeliveryRequired="true">
    <RequestingRole name="Purchase Order Confirmation Sender"
nameID="bt-PIP3A20-role-sender"/>
    <RespondingRole name="Purchase Order Confirmation Receiver"
nameID="bt-PIP3A20-role-receiver"/>
    <!-- No TTAR, no isIntelligibleCheckRequired and no
isNonRepudiationReceiptRequired -->
    <RequestingBusinessActivity
      name="Send Purchase Order Confirmation"
      nameID="bt-PIP3A20-ba-req"
      isNonRepudiationRequired="true"
      retryCount="1"
    >
      <DocumentEnvelope
        name="doc-PIP3A20-PurchaseOrderConfirmation"
        businessDocumentRef="doc-PIP3A20-
PurchaseOrderConfirmation"
        nameID="doc-PIP3A20-PurchaseOrderConfirmation-de"
        isAuthenticated="transient"
        isConfidential="transient"
        isTamperDetectable="transient"
      />
      <!-- No ReceiptAcknowledgement/Exception definitions here -->
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name="xsd-pacifier" nameID="bt-
PIP3A20-ba-resp"/>
  </DataExchange>
```

As mentioned, Use Case 2 involves several Quality of Service specifications for business state alignment, though notably less than Use Case 1. These specifications are described below, with some definitions coming directly from the ebXML Business Process Specification Schema Technical Specification. Please also see [section 8.2: Realization of QoS](#) above for further information on realization of these Quality of Service features.

Receipt Acknowledgement for Business State Alignment:

- There are no acknowledgements specified for this use case. Use Case 1

involves a Receipt Acknowledgement, but Use Case 2 does not specify this requirement. This means that there is no requirement for the Receiver of the PO Confirmation message to send any acknowledgement of receipt (at the PIP protocol layer) back to the Sender.

Security:

- isAuthorizationRequired is not specified in Use Case 2.
This means that there may be no validation that the sending or receiving parties in this business transaction match up with any list of authorized entities.
- Document Envelope settings:
 - isAuthenticated is set to "transient."
This means that authentication is implemented at the communication level. The specific method is determined by the communications protocol used.
If this had been set to "persistent," it would mean that *"the Business Document signer's identity must be verified at the receiving application level"* (cf. ebBP spec.), to assist in verification of role identity.
Note that the configuration option "persistent" is NOT allowed for in the Strict Execution Model.
 - isConfidential is also set to "transient." Transient confidentiality is provided by a secure network protocol, as the message is transferred between two adjacent transport messaging nodes.
In comparison, "persistent" confidentiality would mean that *"the message must remain in encrypted form after it is delivered to the messaging node and will be decrypted only by the authorized application"* (cf. ebBP spec.).
Note that the configuration option "persistent" is NOT allowed for in the Strict Execution Model.
 - isTamperDetectable is set to "transient."
This provides the ability to detect if the information has been tampered with during transfer between two adjacent transport messaging nodes. If set to "persistent," it would provide the *"ability to detect if the information has been tampered with after it has been received by messaging node, between the messaging node and the application. Tamper detection assists in verification of content integrity between participating parties and within a participating party"* (cf. ebBP spec.).
Note that the configuration option "persistent" is NOT allowed for in the Strict Execution Model.

Reliable Messaging:

- isGuaranteedDeliveryRequired is set to "true" for the Data Exchange.
This is a declaration that the sender and receiver must employ only a delivery

channel that provides a delivery guarantee for this business transaction's business message.

Non Repudiation and Non Repudiation of Receipt:

- isNonRepudiationRequired is set to "true" for the RequestingBusinessActivity. This means that the PIP requester must add a fresh digital signature to the business document message so that she cannot claim later not to have sent the message or to have sent a message with different content. The PIP responder, in turn, is assumed to persistently store the business document message together with its signature.
- IsNonRepudiationReceiptRequired is not specified as a requirement in Use Case 2. Since there is no Receipt Acknowledgement in this use case, IsNonRepudiationReceiptRequired will not come into play, as usage of this parameter requires and utilizes the Receipt Ack.

Timing Constraints:

- There is no timeToAcknowledgeReceipt specified in Use Case 2. Since there is no Receipt Acknowledgement required, then a requirement of a time to acknowledge receipt would not come into play here.

retryCount is set to "1" for Use Case 2, which means that the transmission of this message from the sender will only be attempted once before the PIP is considered as failed.

So as shown above, the configuration of the Purchase Order Confirmation PIP is less stringent than in Use Case 1. However it does involve usage of several of the Quality of Service features available, and as such it does involve agreement between the integration partners upon the result of PIP execution, and falls into the Strict Execution Model.

11. Use Case realization (Strict execution context)

This section is non-normative.

For use case 1 of the previous section, prototypic implementations of the control flow of messages are provided as BPEL process definitions:

- **composablePIP3A20-RA-Asynch-v1.0.9.zip** contains sample BPEL process definitions of PIP requester and PIP responder for the **Asynchronous execution with callback** MEP (cf. section 7.2). It uses the RosettaNet composition header structure and a RosettaNet conversion XSD for PIP 3A20. The control flow of the requester and the responder BPEL process is visualized in figures Figure 11 and Figure 12 that show one valid way of integrating the message exchanges between PIP partners and existing business applications (integration with legacy systems is NOT subject to specification of this profile). Existing business applications are explicitly represented by the BE (= backend) roles where both the PIP requester and the PIP responder communicate with their private BE processes. For the PIP responder, the RAC (=ReceiptAcknowledgementCreation service) role encapsulates the functionality for performing validity checks and creating RAs or RAEs. Note that communication between the PIP requester BPEL process and its backend as well as between the PIP responder BPEL process and its RAC and backend are assumed to be safe, i.e., no messages are lost. Moreover, it is assumed that the BPEL primitives used avoid firing two separate transitions of one single state machine at the same time.

The semantics of the process visualizations in figures Figure 11 and Figure 12 correspond to the protocol definitions of section 8.3 (please cf. above).

- XX?yy denotes the event of receiving message yy from role XX
 - XX!yy denotes the event of sending message yy to role XX (and successful delivery because of the strict execution context)
 - XX!yyFail denotes the event of unsuccessfully trying to send message yy to role XX. Note that this is a LOCAL event of the sending process.
 - toTTP and toRA denote the local events that the timeToPerform or the timeToAcknowledgeReceipt timers have run out
- **simplifiedPIP3A20-RA-Synch-v1.0.9.zip** contains sample BPEL process definitions of PIP requester and PIP responder for the **Synchronous execution** MEP (cf. section 7.2). It does not use the RosettaNet composition header structure, but it uses a RosettaNet conversion XSD for PIP 3A20.

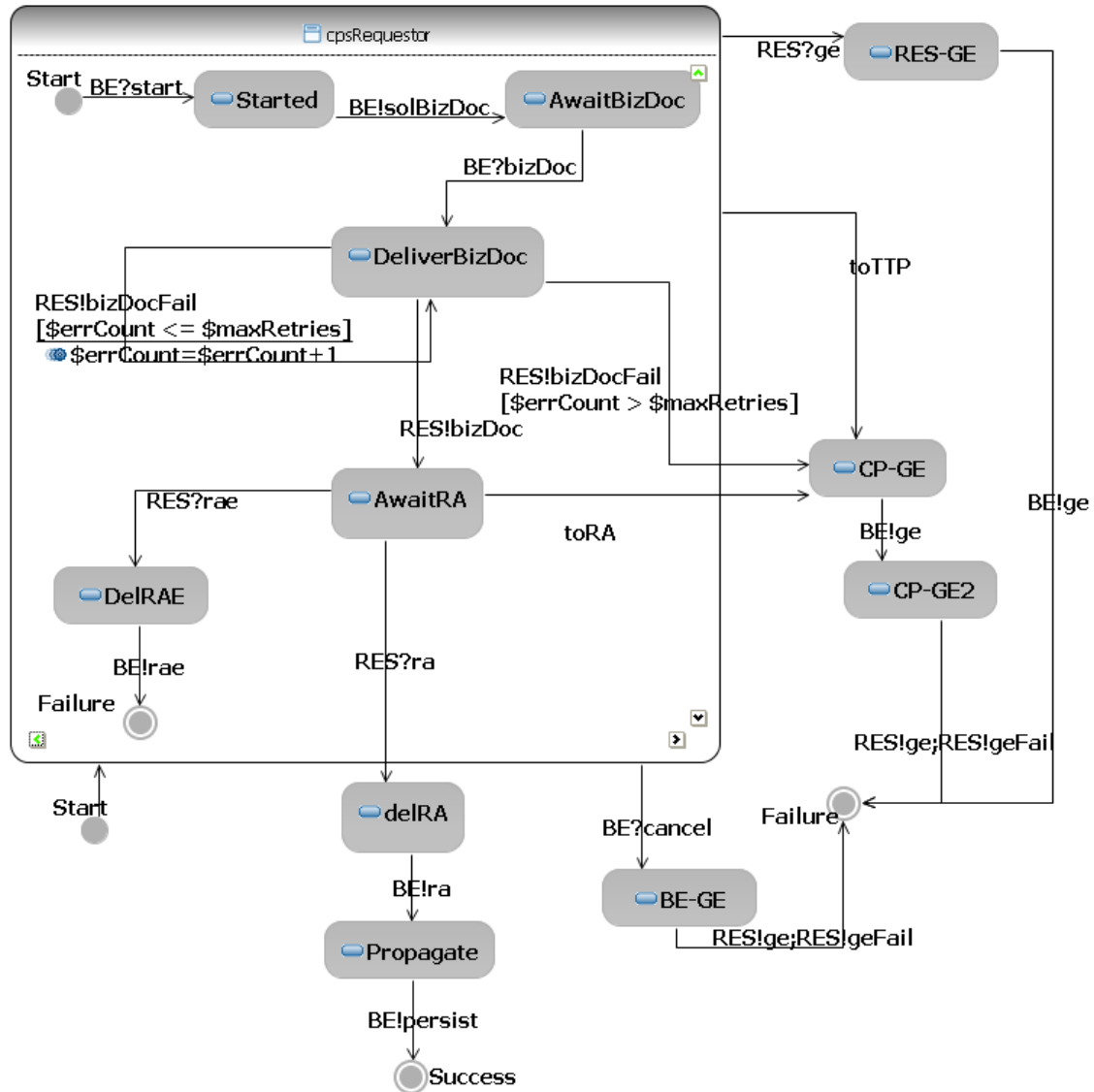


Figure 11: Requester Control Process with Example Backend Integration

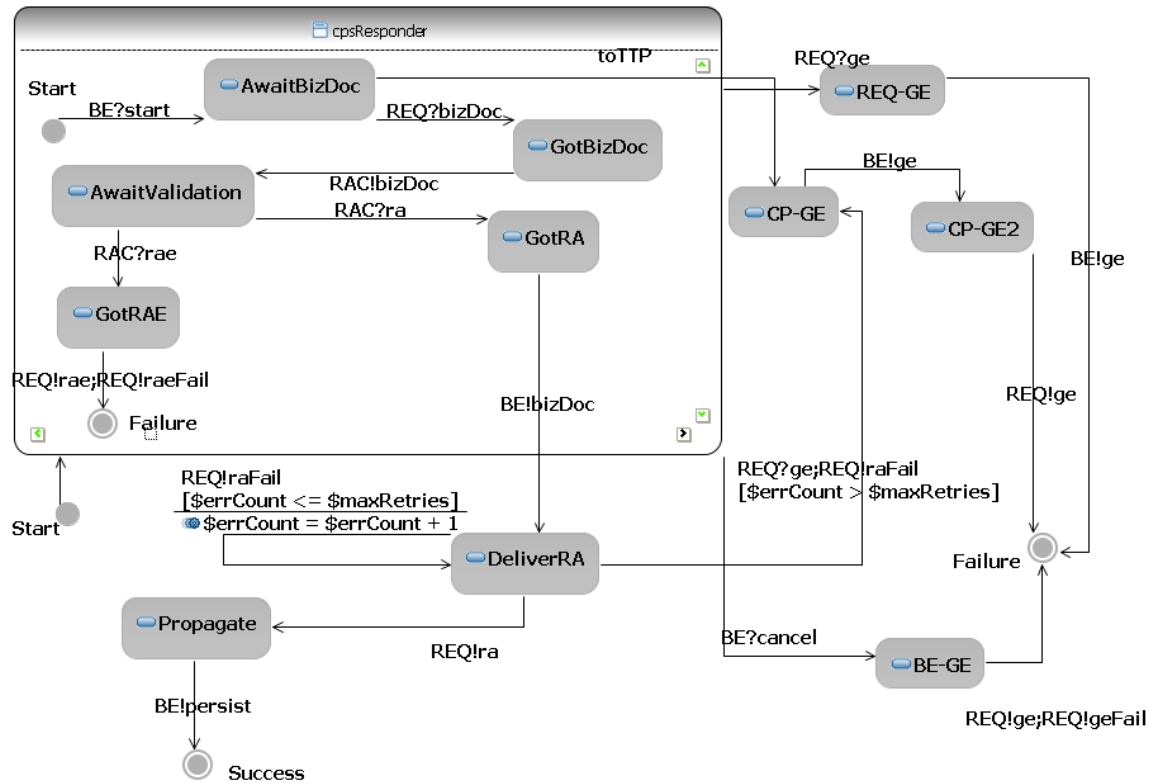


Figure 12: Responder Control Process with Example Backend Integration