`

Multiple Messaging Services (MMS)

Profile for

ebMS 2.0

Document Information	
Name	Multiple Messaging Services (MMS) Profile for ebMS 2.0
Version	Release 11.00.00A
Date Updated	July 25, 2006

Table of Content

Legal Disclaimer

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

Copyright

Trademarks

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

Document Version History

Version	Date	Document History
R11.00.00A	24 July 2006	Release membership approved specification to community at large.

MMS ebMS Team

Members	Company
Jacques Durand	Fujitsu
Dale Moberg	Cyclone Commerce
Peter Hawtrey	ePSA / Tag Tools
Nikola Stojanovic	RosettaNet / GS1US
Wan Norhazlina	RosettaNet (on loan from Importek)
SooChin Yeoh	RosettaNet
Annabelle Kongkeo Marlow	RosettaNet
Ken Siu	Centre for E-Commerce Infrastructure Development (CECID) University of Hong Kong
Deepak Bhargava	Cisco Systems
Elham Ghassemzadeh	Cisco Systems
Jim Kao	Cisco Systems
John Voss	Cisco Systems
Mir Baqar	Cisco Systems
Gary Binder	DHL
Martin Evans	Formfill
Stewart Witchalls	Formfill
Kenji Nagahashi	Fujitsu
John Cartwright	Intel
Frederic Herzer	Motorola

Maggie Caligiuri	Motorola
Jeff Hutchins	Oracle
Wellwin Kwok	Centre for E-Commerce Infrastructure Development (CECID) University of Hong Kong
Mark Schenecker	SAP
Li Xueheng	Sterling Commerce
Ron Boutell	Sterling Commerce
Suresh Damodaran	Sterling Commerce
Terence Thambi Rajah	Sterling Commerce
Bill Wray	Tyco Electronics
Gary Sheetz	Tyco Electronics
Bala Yamandra	webMethods
David Smiley	webMethods

1 Introduction

1.1 Document Conventions

This document occasionally uses terms that appear in capital letters. When the terms "MUST", “REQUIRED”, “SHALL”, "SHOULD", "RECOMMENDED", “MAY”, “OPTIONAL”, "MUST NOT", “NOT REQUIRED”, “SHALL NOT” and "SHOULD NOT" appear capitalized, they are being used to indicate particular requirements of this profiling specification. The meaning of these terms is to be interpreted as defined in [RFC2119].

1.2 General Intent and Scope

1.2.1 Intent

RosettaNet implementations currently require users to buy a messaging system capable of running the RosettaNet Implementation Framework (RNIF). Although such systems are robust and widely adopted for XML payloads in the high-tech industry, such RNIF messaging systems are not commonly used by other vertical markets. As a result, companies who support both the high tech industry and other verticals are forced to support more than one messaging standard for e-business transactions. To alleviate the problem, RosettaNet created a Multiple Messaging (MMS) initiative that included three additional messaging systems, ebMS, AS2 and WS-I.

The purpose of this document is to recommend to users and developers how best to use an ebMS message handling system to transport RosettaNet PIP business messages between trading partners. Including ebMS in this way will add another messaging option for trading partners transporting RosettaNet business payloads and may lower the infrastructure investments required to exchange RosettaNet payloads across trading networks that employ ebMS. It is hoped that adding ebMS may also broaden the reach of RosettaNet by encouraging PIP implementations across ebMS based vertical industry and government boundaries.

1.2.2 Scope of This Document

This document is concerned with the profiling and the configuration of an ebXML framework, for carrying RosettaNet PIPs according to the requirements identified in the MMS - Abstract Message Definition (AMD) document.

The approach we have taken is to focus on ebMS 2.0, include part of the CPA and limit choreography to the lowest level covering one action PIPs.

- **ebXML version**: To focus on ebMS 2.0, although ebMS 3.0 will be a profiling target in a next release. It is expected that the profiling done for ebMS 2.0 can

be largely leveraged by ebMS 3.0 profiling. Some desirable features that are likely to be natively supported by ebMS 3 (message pulling, bundling of payloads…) could be emulated to some extent in ebMS 2. However the approach has been to not do this. The ebXML context for this profiling document includes BPSS 1.x ((in RN PIPs: BPSS 1.01, though a customized version – validate against 1.01 at least. Some references will be made to ebBP 2.0 when appropriate), CPPA 2.0 / CPPA 2.1.

- **Agreements**: although ebMS is the target, the intent of this profiling goes beyond just wire interoperability and addresses some agreement aspects (CPA). Defining CPA templates or guidelines is the best way to represent the out-of-band agreement required for a practical deployment of PIPs. Only the part of the CPA that relates to messaging and maps to PIP definition data, will be profiled.

- **Choreography**: The approach is here to only cover the lowest level of choreography, involving signals possibly associated with only one action message. Any choreography that involves more than one action message (e.g. as in two-action PIPs) is out of scope of this profile specification.

1.3 General Methodology

- The Trading Partner Profiles (TPP) and resulting Abstract Trading Partner Agreement (ATPA) is a good starting point for users. Although the core of the ebXML profiling described here is defined solely based on mapping RNIF features into ebXML features (via the MMS-AMD [AMD] requirements), the ATPA represents parameters that need to be defined in order to complete a user-specific profiling of ebXML for a PIP deployment.

- From the TPP info, a CPP (Collaboration Protocol Profile) template can be filled for each partner, and a resulting agreement (TPA) can be mapped to a subset of the CPA. However, the suggested approach is for a business entity to directly define a partially-filled CPA with its capabilities and communication requirements (a “CPA profile”), then share this CPA profile with its business partner(s) who will complete it. The resulting document is a CPA instance.

- This CPA instance will be somehow orthogonal to PIPs: the deployment of several PIPs may share the same CPA instance. Conversely, several instances of the same PIP may use different CPAs, based on requirements that are specific to the nature of the document contents and other business considerations.

- This document describes **profiling rules** for CPA and ebMS. These rules, when applied to data that is specific to business partners (TPA) and specific to targeted PIPs, will define specific **messaging profiles**.

- ebMS can be used with either BPSS or CPPA, and so both BPSS and CPPA can also be used when ebMS is used for RosettaNet. No attempt will be made to produce a complete profile for CPPA or BPSS for RosettaNet in this document. However, it will occasionally be explained what CPPA or BPSS features would

need to be in a CPPA or BPSS that governed ebMS messaging when used for RosettaNet. These features can be understood as configuration input for the ebXML MSH mode of operation, as they may affect the MSH behavior without necessarily affecting the message header.

Some general rules of ebMS profiling:

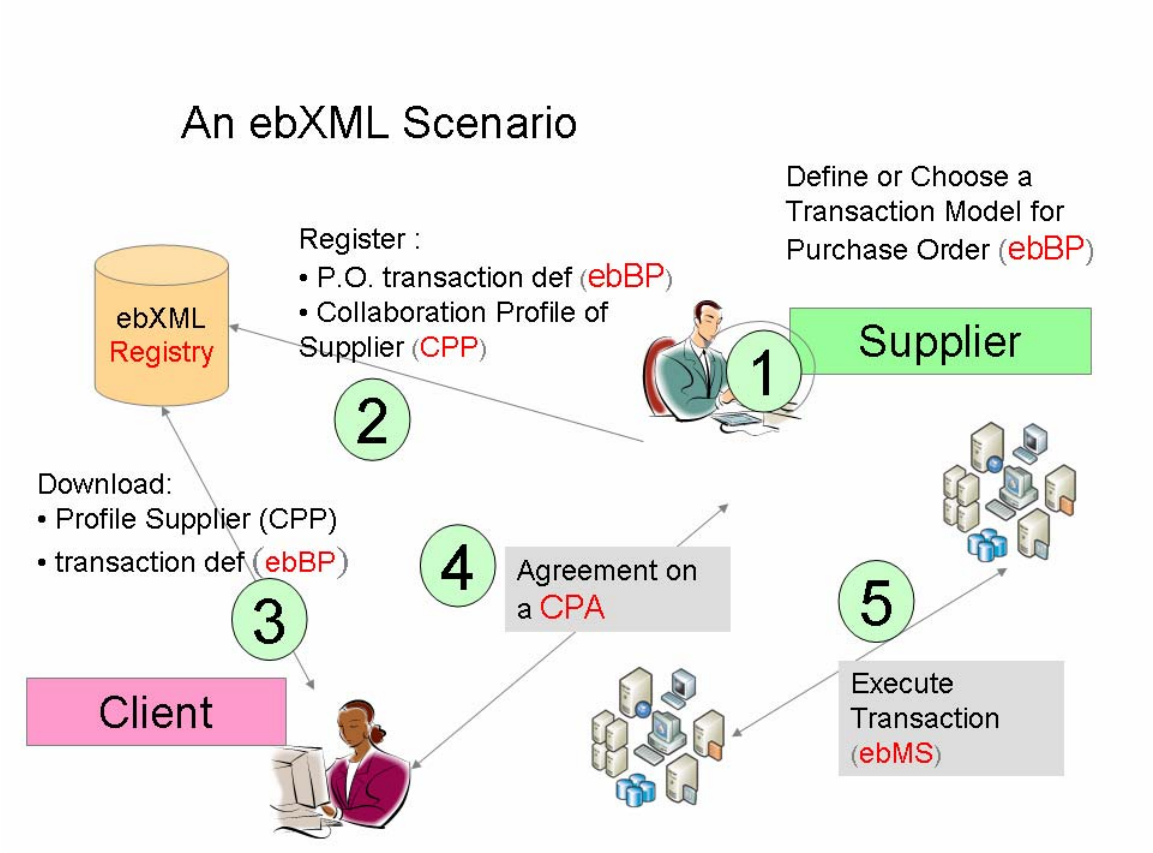
- The proposed profiling in this document does not make use of extensibility points in the ebMS header, because it is unlikely that current MSH implementations can process these, even if they tolerate them.
- The ebMS header will fulfill the functions of the RNIF Delivery header, although it does not contain all the information present in the Delivery header. Some elements of the ebMS header will also map to elements from the Service header.
- Clearly there is more data in Standard Business Document Header (SBDH) and/or RNIF Service Header than can be represented in ebMS headers (without using extensions). In case this data needs be preserved, then related XML parts need to be added in the payload, transparently to ebMS processing.

2 ebXML Messaging Overview

2.4 The ebXML Framework

The ebXML specifications support the exchange of business messages required to conduct electronic trading relationships between business partners. These capabilities logically separate but allow coordinated use of five key technologies important for eBusiness:

- Communicating data in common terms using a defined methodology (Core Components)
- Defining business processes and assembling business transactions (ebXML Business Process Specification Schema, ebBP)
- Providing secure and reliable transport (ebXML Messaging Service [ebMS])
- Registering and making available key eBusiness artifacts and services (ebXML Registry Services [ebRS] and Registry Information Model [ebRIM])
- Providing a technical configuration contract between business partners (Collaboration Protocol Profile and Agreements [CPP/CPA])



The business case and requirements that prompted the development of the set of ebXML specifications were to:

- Provide a migration path for and leveraging of EDI-compatible technologies.
- Develop openly accessible technologies for Small-Medium Enterprises whether in a managed or non-managed environment.
- Enable all supply-chain partners using XML technologies.
- Provide an integrated eBusiness approach focused on interoperability needs, while maintaining loose-coupling to backend systems.

The heterogeneous nature of eBusiness transactions require a flexible infrastructure and architectural framework that can support service calls (catalog status requests) and more complex document exchange (offers and acceptance).
The business document processing was decoupled from the messaging layer. The consumer of these documents may be a service, a business process instance, or middleware or business application interested in the business document contents. Such documents cannot be directly associated with an application service in a predefined way. The coupling between the messaging system and the consumers of these messages must be supported in an adaptable way.

2.5 ebXML Messaging

The ebXML Messaging Service technical specification defines a communications-protocol neutral method for exchanging electronic business messages. It defines specific enveloping constructs that support reliable, secure delivery of business information and permits messages to contain payloads of any format type. This versatility ensures legacy electronic business systems employing traditional syntaxes (i.e. UN/EDIFACT, ASC X12, or HL7) can leverage the advantages of the ebXML infrastructure along with users of emerging technologies. For example, the ebMS may be used with the Collaboration Protocol Profile and Agreements (CPPA) and Business Process Specification Schema (ebBP).

Messaging Overview

ebMS was developed to provide the secure and reliable transport of electronic messages, while enabling the transition from legacy to new and emerging technologies. The ebMS protocol supports any communication protocol, and provides bindings for HTTP and SMTP**.** Multiple
message exchange patterns including push and pull that map to different types of business transactions.

Status of ebXML Messaging Standards

The ebMS v2.0 OASIS Standard was approved in August 2002, and, in May 2004, submitted and accepted an ISO standard, ISO/TS 15000-2. The v3.0 working draft is under development.

In v2.0, the current feature set for ebMS includes:

- Packaging

- SOAP envelope extensions
- Error handling
- Security
- Synchronous responses
- Reliable messaging (at-least-once, at-most-once, ordering)
- Message status service
- Multi-hop

The ebXML Messaging Service v3.0 working draft is seeking to express compatibility to or use with emerging web services standards or specifications. Seeking to leverage the ebXML capabilities and those of emerging standards and specifications, the ebMS will define how technologies such as WS-Reliability and WS-Security, for example, may be used and the role of WS-I Basic Profile conformance. Composability with WS-Addressing is also under consideration. The ebMS technical specification v3.0 will support:

- Web services protocols for security, reliability and addressing.
- Low-level payload processing (payload services)
- Message exchange patterns that map to business transactions.
- Connectivity constraints (occasional connectivity, firewall restrictions, ebMS-lite clients, etc.)

3 Trading Partner Agreement

3.6 TPA and Infrastructure Deployment Parameters

This table shows the relationships between TPA and ebXML components. It shows how elements of a TPA relate to the ebXML components. A dependency in the table indicates that some features of the ebXML component are concerned by the TPA item, or even further, that the feature must comply with the profiling described in this document.
The elements of a TPA that do not directly affect the standardized features of an ebXML component may still concern this component, but its impact will be at implementation, administration or deployment level, i.e. will affect the operational aspect, which is outside the scope of this profiling guideline.

- An “F” means that the answer to the question will affect features that are specified in the related standard (ebMS, CPPA or BPSS). In other words, a compliant implementation to these standards explicitly supports answers to this question.

- An “X” means that depending on the user answer to the question, the usage of this ebXML component will be affected in a way that must comply with the profiling defined in this document. In other words, you need to be aware of this profiling in order to implement the answer in a compliant way.

- No mark means no direct effect or dependency on features that are specified in the related standard. The answer to the question becomes more a product implementation or deployment issue (how well this is handled depends more on additional product features than on conformance to the standard.)

TPA Item	ebMS	CPPA	BPSS (no profiling defined here)
General Operation Parameters			
Specify the standard you will use to identify trading partners. (DUNS, GLN, other authorityname)	FX	FX	
What is the maximum volume of messages you expect to exchange with any specific trading partner? What is the average?			
What is your peak time interval?			
What is the average size of the message during the peak time interval?			
How many messages do you receive and send during the peak time interval?			

Will you use peer-to-peer routing or will you be using a messaging service (MS) (temporary message store)?			
Will you operate via a message Hub, or directly point-to-point?	FX	F	
If using a Hub, which message data will be used for doing routing ?	FX		
Which of these Message Exchange Patterns will you be using? One-Way, Request-Response, Notification, Solicit-Response	FX	FX	F
Will you need to correlate messages from a single PIP instance, or will you need to correlate messages from several PIP instances, as belonging to a same long-lasting conversation (e.g. for monitoring purpose)	FX	FX	F
Will you be using a specific registry? If Yes, please specify			
Will you be defining specific roles for each party of each business process?	FX	FX	F
Indicate the level of Management Services you will require for your messaging system	F		
Do you need to track messages based on their participation in complex business processes?	FX (maybe)		
Do you need status services?	F (maybe)		
Do you need remote management?			
Do you need monitoring?	FX (maybe)		
Do you need BI support?			
Do you require disaster recovery?			
Do you require Debugging capabilities?			
Indicate the level of Connectivity you will have with the Internet			
Are you occasionally connected (dialup with modem?)	F	F	
Are you permanently connected with a permanent IP address (T1, DSL)?	F	F	
Are you permanently	F	F	

connected without a permanent IP address (Cable)?			
Indicate the Security Services you will require for your messaging system			
Do you need the headers to be encrypted?	FX	FX	
Do you need the payload to be encrypted? Using which method? (S/MIME, XML Encryption)	FX	FX	
Do you need non-repudiation of origin?	F	F	
Do you need non-repudiation of receipt?	FX	FX	
If need non-repudiation of receipt, is it required to have a digest of the acknowledged message in the acknowledgement, or is a simple reference to MessageID sufficient?	FX		
Do you need to digitally sign your messages (i.e.:X.509)?	FX	FX	
Do you want to transport level encryption? (i.e.: TLS/SSL)?	F	F	
Do you require timestamp of your messages?	F		
Do you subscribe to authority domains? (i.e.: DUNS/GLN/EAN-UCC/DUNS+4/Vertical)	FX	F	
Indicate the Reliability you will require for your messaging system			
Do you need guaranteed message delivery? (include ACK signals)	FX	FX	
Do you need de-duping?	FX	FX	
Do you require Ordered Delivery?	FX	FX	
Do you require a Manifest?	F		
Do you require expiration control?	F		
Indicate what you will need to compress in your messages			
Do you need to compress	F		

headers?			
Do you need to compress payload?			
Do you need to compress attachments?			
Indicate the Error Handling you will require for your messaging system			
Do you need Message Service error handling (too many retries, etc.)?	F	F	
Do you need Message Content error handling (invalid message, etc.)?			
Do you need out of band error notification?			
Indicate the Payload Capabilities you will require for your messaging system			
Do you need Globalization/I18?	F		
Do you need attachments?	F		
Do you need payload validation?			
What type of payload will you be exchanging, ASCII, Binary or both?			
Will you need to support different versions of PIPs?			
Indicate the requirements imposed by integration with existing software			
Do you need to preserve (some) previous message header structures (not native to ebMS) as is, so that you can reuse back-end binding technology?	FX		
Do you need to move message header data into your back-end?	FX		

3.7 CPP and CPA Profiling

Part of the TPA options of the previous section will map to the CPA, or to the Collaboration Protocol Profiles (CPP).

Business partners may define CPP that represent their capabilities and roles they can assume. Another approach is for a business party to directly start by defining a CPA template that this party will pre-fill with its own data, and that it will communicate to its partners for them to complete. The partially created CPA (or CPA template) will narrow the options that a CPP would offer, down to a very specific way under which this party wishes to interoperate. This is the approach suggested here.

A party may define a few of such CPA templates that express different modes of connectivity, different roles and collaborations and different QoS attributes. The reason for doing so is that its business partners may have different profiles. For example, three Trading Partner profiles have been identified in [AMD]. A CPA template may be intended for each one of these partner profiles.

There is no 1-to-1 correspondence between CPAs and PIP definitions.

- Instances of the same PIP may need to be executed using different CPA templates (or different CPA instances from the same template) depending on which types of partners are involved, and which QoS or execution context is required.
- Instances of different PIPs or TPIR-PIPs may use CPAs that derive from the same CPA template, yet these CPA instances will not be shared across different PIP definitions, since the PIP (or TPIR-PIP) identity appears as attribute of the CPA instance.

The profiling and definition of CPA data (both instance and template) can be facilitated using a set of forms that we define in Appendix A.

4 Message Description

4.1 General approach for ebMS 2.0 profiling

4.1.1 ebMS message headers and RosettaNet message headers

RNIF Headers:

The existing RNIF headers are dealt with in the following way:

- the RNIF Preamble header is not supposed to appear anymore in the ebMS message.
- the RNIF Delivery header is not supposed to appear anymore in the ebMS message. It is replaced by the ebMS header. Although not all the information encoded in the Delivery header will be translated into the ebMS header, this header is not relevant anymore in the ebXML context.
- the RNIF Service header may still be present in the ebMS message, in case it is needed for binding a message to existing PIP software that needs to be reused with ebXML. Some but not all of its elements map to the ebMS header. The Service header may be preserved as a separate attachment in the ebMS payload. However, in case of conflict between data in Service header and analogous data in ebMS header, the ebMS header will prevail as long as the messaging transfer is still in progress (this includes routing in multi-hop environments).

Standard [Business] Document Headers (SBDH):

Not all information in the SBDH will map to ebMS headers. In case where an SBDH structure is present and used in the binding of the message with backend processes, it is recommended to keep the SBDH in the payload. However, in case of conflict between data in SBDH and analogous data in ebMS header, the ebMS header will prevail as long as the messaging transfer is still in progress (this includes routing in multi-hop environments). There are two options to consider:

1. The PIP document is defined using XML schema. In this case the SBDH is bound to the PIP document, and both represent a single MIME part (SOAP attachment). The ebMS2 message has then two MIME parts: (a) the SOAP envelope (ebMS2 header), (b) the payload (SBDH + PIP).
2. The PIP document is defined using DTD. In this case, no SBDH instance is required (additional meta-data that may be needed and that is not in ebMS headers, is supposed to be found in Service header – which can be preserved as a separate MIME part in the ebMS payload if needed -, and FromRole + ToRole elements of PIPs.)

ebMS header extensions:

Although the ebMS header is extensible, extension elements should not be used to store data from previous headers (Delivery, Service) that could not be mapped. The reason for this is that MSH implementations are not required to process these extensions. Instead, in case the non-mapped information of legacy headers must be preserved in the message, these headers should be included as is in the payload (e.g. as attachments). A deployment that conforms to this messaging profile MUST NOT use such extensions.

Integrating with existing infrastructures:

In order to preserve the ability to reconstruct the former RNIF structure (minus preamble and delivery headers), in case some users consider this a good integration approach with existing systems, the Service header MAY be included as a separate MIME part in the ebMS2 message, i.e. as an additional SOAP attachment element of the ebMS payload.

4.1.2 Payload bundling and message batching

- **Bundling of payloads** is the grouping of several payloads within the same message, along with the ability to process each payload differently, as if they were carried in different messages. The message header is not repeated: it is common to all the payloads in the bundle. In case the payloads are not intended to share the same header elements (e.g. not intended to the same party, or same business process) then they cannot be bundled in ebMS 2.0. Although it is possible to group several payloads in the same message as attachments, an ebMS 2.0 MSH would not be able to process these differently as the same set of header elements would apply to all of them.

- **Message batching** is the ability to nest several ebMS messages (including their individual headers) within the same ebMS message, and on the receiver side to process them individually as if they had been received separately. The difference with payload bundling is that a set of well-formed ebMS messages would be bundled, not just their payloads. On receiver side, after un-packaging, each one of these messages is processed individually as if it had been received separately. This can be done by compressing each individual ebMS message as a MIME part. Although this can be implemented by using compressed attachments, ebMS 2.0 does not support batching of messages as a specified MSH function.

For these reasons, neither message batching nor payload bundling is recommended when profiling of ebMS 2.0.

4.2 ebMS Header Profiling

The tables below are borrowed from the Deployment Profile Template 1.1 for ebMS 2.0, [DPT-ebMS2] a document guide for deploying ebMS that has been developed by the ebXML Implementation, Interoperability and Conformance OASIS TC. It is recommended that business partners make use of the full document to define their own deployment profile. Only the subset of this document that is affected by the MMS profiling is used here.

4.2.1 Profile Requirement Item eb:PartyId

Specification Feature	Header elements: eb:MessageHeader/eb:From/eb:PartyId eb:MessageHeader/eb:From/eb:PartyId/@type eb:MessageHeader/eb:To/eb:PartyId eb:MessageHeader/eb:To/eb:PartyId/@type
Specification Reference	ebMS 2, section 3.1.1.1 “PartyId Element”
Profiling	One instance of PartyId (in case several exist) must have as value either a DUNS (or DUNS+4) or Global Location Number (GLN). Both should not be found at the same time under the same From or To element. When several PartyId are present, the one above should be the first PartyId element. It is allowed to have additional PartyId elements in eb:From or in eb:To (they just need to have different @type values) • PartyID values MUST comply with ISO 6523 values, when applicable. That includes DUNS and GLN identifiers. • PartyID type attribute MUST be used to represent the Domain name and ICD (International Code Designator) according to section 24 “PartyID” of the CPPA V2.1 specification, found in: http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_1.pdf (April 2005) More generally: • the type value of the type attribute MUST be a URN. If the type attribute is present, then it provides a scope or namespace for the content of the PartyId element. • if the type attribute is not present, the content of the PartyId element MUST be a URI that conforms to [RFC2396]. If an abbreviated name is described in the item titled “Name of Coding System” within the ICD list, it should be used, followed by the ICD value. Example: <tp:PartyId tp:type=" urn:oasis:names:tc:ebxml-cppa:partyid-type:D-U-N-SNumber:0060"> 123456789</tp:PartyId> Where “0060” is the ICD value of D-U-N-S Number.

Alignment	• MUST map to (RNIF, Delivery Header) element: PartnerIdentification / GlobalBusinessIdentifier. • MUST map to (Standard Bus. Doc Header - SBDH) element: PartnerIdentification (choice of Duns / Duns+4), when applicable. • MUST map to ebXML CPPA 2.0 or 2.1 element: PartyInfo/PartyId, when used. • Value and type MUST conform to ISO 6523 when applicable, and the type attribute to section 24 “PartyID” of the ebXML CPPA V2.1 specification.
Test References	

4.2.2 Profile Requirement Item eb:Role

Specification Feature	Header elements: eb:MessageHeader/eb:From/eb:Role eb:MessageHeader/eb:To/eb: Role
Specification Reference	ebMS 2, section 3.1.1.2 “Role Element”
Profiling	Role Name of the partner in this business transaction (in Partner Role Description of the PIP) **Example:** <eb:To> <eb:PartyId eb:type=" urn:oasis:names:tc:ebxml-cppa:partyid-type:D-U-N-SNumber:0060">myDUNS</PartyId> <eb:Role>Seller</eb:Role> </eb:To>
Alignment	• MUST map to (RNIF, Service Header) element: from{to}Role/GlobalPartnerRoleClassificationCode • MUST map to ebXML BPSS: the role value maps to the corresponding BinaryCollaboration/Role/@name in BPSS 1.* or ebBP 2.0 definition, when used. • MUST map to ebXML CPPA 2.0 or 2.1 element: CollaborationRole/Role/@name, when used.
Test References	

4.2.3 Profile Requirement Item eb:Service

Specification Feature	Header element: eb:MessageHeader/eb:Service
Specification Reference	ebMS 2, section 3.1.4 "Service Element"
Profiling	This value MUST be the same as the one used in the BPSS instance for the PIP. Its format must be: **"urn:rosettanet:specification:interchange:PIP" +<alphanumeric name of PIP> + ":xml:bpss:"+ <PIP VersionIdentifier>.** The Service element MUST have same value for all messages involved in a single PIP or TPIR-PIP (whether it is an Action, Confirmation or Signal message). **Example:** If in BPSS: nameID="**urn:rosettanet:specification:interchange:PIP7C7:xml:bpss:v11_00**" version="**V11.00**" **In ebMS header:** <eb:Service> urn:rosettanet:specification:interchange:PIP7C7:xml:bpss:v11_00</eb:Service>
Alignment	• MUST map to ebXML BPSS 1.* or ebBP 2.0 element when used: ProcessSpecification/@uuid or if not present, ProcessSpecification/@NameId • MUST map to ebXML CPPA 2.0 or 2.1 element, when used: Service/@name
Test References	

4.2.4 Profile Requirement Item eb:Action

Specification Feature	Header element: eb:MessageHeader/eb:Action
Specification Reference	ebMS 2, section 3.1.5 "Action Element"

Profiling	In case of an Action message, the value MUST be consistent with Global Business Action Code (camel case version of this value). In particular, it MUST map to (RNIF): **<Business Document Name> + "Action",** or in case of 1-way PIP and a responding activity, specify Activity Name**: <Responding Business Activity>.** In case of a RosettaNet signal, the value MUST be consistent with the following: If Signal is positive (ReceiptAcknowledgment): **Value= "ReceiptAcknowledgment"** If Signal is negative (Exception): **Value= "Exception"** **Examples:** <eb:Action>PurchaseOrderRequestAction</eb:Action> <eb:Action>PurchaseOrderConfirmationAction</eb:Action> <eb:Action>SemiconductorTestDataNotification</eb:Action>
Alignment	• MUST map to ebXML BPSS 1.* element when used: **RequestingBusinessActivity/@nameId, RespondingBusinessActivity/@nameId** • Map to ebXML BPSS 2.0 element when used: BinaryCollaboration//@name or BusinessCollaboration/@name, Or, more precisely, the no-space version of these values.
Test References	

4.2.5 Profile Requirement Item eb:ConversationId

Specification Feature	Header element: eb:MessageHeader/eb:ConversationId
Specification Reference	ebMS 2, section 3.1.3 "ConversationId Element"
Profiling	It is RECOMMENDED that ConversationId represents the PIP instance ID value, i.e. has same value for all messages related to the same PIP instance. In other words, messages from the same PIP instance MUST have same ConversationID, and it is recommended that this ConversationID be unique to this PIP instance (not shared with other PIP instances).
Alignment	• MUST map to Standard Business Doc Header (SBDH**)** element when applicable**: RequestingDocumentInformation / BusinessProcessInstanceIdentifier**

	• MUST map to (in RNIF Service header) element: `ServiceHeader/ProcessControl/pipInstanceId/InstanceIdentifier`
Test References	

4.2.6 Profile Requirement Item eb:RefToMessageId

Specification Feature	Header element: eb:MessageHeader/eb:MessageData/eb:RefToMessageId
Specification Reference	ebMS 2, section 3.1.6.3 "RefToMessageId Element"
Profiling	As a reminder, it MUST be used only for: (1) relating business signals messages to action messages, by referencing the ebMS ID. (2) Relating a business response to a business request. Every message involved in a PIP instance MUST refer to another previous message of this instance (except for the initial message of the instance, which MUST NOT have a RefToMessageId element.) If several PIP instances must be correlated, this MUST NOT be achieved by this value (the first message of a PIP instance MUST NOT refer to another PIP).
Alignment	• MUST map to RNIF Service header element, in the sense it plays a similar role: `ServiceHeader/ProcessControl/ActivityControl/MessageControl/inReplyTo/messageTrackingID`
Test References	

4.2.7 Profile Requirement Item eb:MessageId

Specification Feature	Header element: eb:MessageHeader/eb:MessageData/eb:MessageId
Specification Reference	ebMS 2, section 3.1.6.1 "MessageId Element"
Profiling	Used for uniquely (globally) identifying a message (either signal or action). Normally, this identifier is automatically generated by an MSH, and out of control from applications. (However, an MSH provides visibility to applications on this value, so that an application can use it for referencing (see eb:RefToMessageId).
Alignment	MUST map to RNIF Delivery header, in the sense it plays a similar role: `DeliveryHeader/messageTrackingID.InstanceIdentifier`.
Test References	

4.2.8 Profile Requirement Item eb:CPAId

Specification Feature	Header element: eb:MessageHeader/eb:CPAId
Specification Reference	ebMS 2, section 3.1.2 "CPAId Element"
Profiling	See Section "CPA Profiling" in Appendix A, for recommended profiling.
Alignment	
Test References	

4.2.9 Profile Requirement Item eb:Manifest

Specification Feature	Header element: SOAP:Body/eb:Manifest/eb:Reference SOAP:Body/eb:Manifest/eb:Reference/eb:Schema
Specification Reference	ebMS 2, section 3.2 "Manifest Element"

Profiling	The eb:Reference child element MUST comply with the following (i.e. the following attributes and child elements MUST be used): Attribute Xlink:type="simple". Attribute Xlink:href: MUST contain a content id (URI scheme: "cid") and not a URI that cannot be resolved locally. Example: xlink:href="cid:payload-1" Element eb:Schema: This element SHOULD be present when the eb:Reference element is referring to the service content part (main XML document) of a PIP payload, or is referring to an XML RosettaNet signal. When present it MUST use an URN identifying the schema or DTD that applies to the part. The schema URN identifier MUST comply with **[RN-NameSpaces].** Example for a PIP service content with XML schema: `urn:rosettanet:specification:interchange:PIP3A4PurchaseOrderRequest:xsd:schema:1.0` When a legacy RNIF header (such as Service header) is included in the message, it must be added as a single attachment. The eb:Reference element SHOULD contain an eb:Schema element to identify it, which conforms to **[RN-NameSpaces].** Example for a Service header: `urn:rosettanet:specification:system:ServiceHeader:dtd:schema:2.0` NOTE: The use of an XLINK processor should not be required.
Alignment	
Test References	

5 Message Processing

5.1 Packaging

The packaging of the message headers and payloads, including security headers, follows the ebMS 2.0 specification. It is automatically implemented by conforming ebMS MSH implementations.

5.2 Un-packaging

The un-packaging of the message headers and payloads, including security headers, follows the ebMS 2.0 specification. It is automatically implemented by conforming ebMS MSH implementations.

6 Message Exchange Patterns and Representation

This section only describes the lowest level of MEPs involving only one action message, and some RosettaNet signal message(s). Higher level choreographies, such as those involving more than one action message (including two-action PIPs), are out of scope of this message profiling.

In the following, a clear distinction must be made between ebMS signal messages and RosettaNet signal messages.

- An ebMS signal message has no significance or visibility to the user layer: it is an error message or a reliability acknowledgement that is for the exclusive usage of the ebMS protocol. It would not be visible in any message choreography that RosettaNet defines.
- A RosettaNet signal message (receipt, exception) is, form an ebMS perspective, like any other application level message. ebMS does not handle these any differently from action messages. The signal content is considered as any other application payload.

Note: It is however possible to piggyback ebMS acknowledgements or errors on application messages, merging into one message what could have been considered as two separate messages (an ebMS signal and an application message.)

6.1 MEPs in ebMS 2.0

Two basic MEPs are supported in ebMS 2.0. :

- **One-way Push MEP**: Supports the sending of one message (PIP signal or action) as ebMS message, initiated by the sender. A single ebMS message is exchanged in this MEP. In case a request-response protocol such as HTTP is used, no application-significant ebMS message is returned over the response leg of the MEP. In this mode of messaging, every message of a PIP instance would be sent over a separate instance of the One-way Push MEP. This is the default mode of operating in ebMS 2.0. ***syncReplyMode*** in the CPA must be either absent, or set to *mshSignalsOnly***.** In the latter case, only ebMS signals can be sent back as response (not RosettaNet signals.) In the context of this profiling specification, this MEP can be used for sending:
  - A RosettaNet Action message (with possibly an ebMS signal on the response leg)
  - A RosettaNet Signal message (with possibly an ebMS signal on the response leg)

- **Request-response MEP**: Supports the sending of one message (PIP signal or action) as ebMS message, initiated by the sender, and the sending of an application-level response message as an ebMS message. Two ebMS

messages are exchanged in this MEP, over the same request-response of the underlying protocol. This requires a request-response underlying protocol. ***syncReplyMode*** in the CPA must set to *signalsAndResponse* or *responseOnly*. This requires the first ebMS message to use syncReply mode. In the context of this profiling specification, this MEP can be used for sending:

- A RosettaNet Action message (request) followed by a RosettaNet Signal message (response).

Pulling of messages is not supported in ebMS 2.0 but will be in ebMS 3.0. Although this feature could be emulated on top of ebMS 2.0, the recommendation is here to not use it with ebMS 2.0.

An ebMS MSH does not require more configuration information other than what is the type of MEP a message being sent is participating in. This is concretized by the SyncReplyMode value, defined in the CPA associated with that PIP, for a particular trading partner. Any choreography at a level higher than those described above, is out of scope of ebMS messaging and will need to be controlled by a layer above messaging. However, in any case, some message correlation is apparent in the ebMS message header: ConversationID and RefToMessageId must follow the profiling recommendations in Section 4.2.

6.2 Handling of Receipts

We consider here the basic message sequence of sending an Action message, and getting back a Receipt Acknowledgement or an Exception. As ebMS 2 provides a reliable messaging feature, it appears that there is significant overlap of purpose with the Receipt Acknowledgement signal in RN:

- A retry mechanism is specified in RN, triggered by not receiving a Receipt Acknowledgement in time. It is controlled with a RetryCount parameter, and Time to Acknowledge.
- ebMS 2.0 has a similar mechanism of retries, until an acknowledgement is received. A maximum number of retries, as well as, a retry interval, are specified.

The RN Retry mechanism can be largely supported by ebMS 2.0 reliability, except for two aspects:

- the meaning of a Receipt Acknowledgement usually goes beyond message reception, to include document validation (grammar level). The ebMS acknowledgement does not have this semantics.
- Receipt Acknowledgements are meaningful for non-repudiation of receipt. The signed-ack in ebMS is not a substitute. It does not include a digest of the original message.

The recommended profiling is as follows:

When non-repudiation of receipt is not required	Do NOT use the Receipt Acknowledgement (positive) signals. Instead, use the reliable messaging feature. The mapping of the respective parameters of these features has been described in Appendix A.
	In case of invalid payload: only then would an exception message (type: Receipt Acknowledgement Exception) be sent back, as the result of a validation check occurring at higher level than the messaging layer. The absence of such a signal tells the sender that the payload was valid. Note: a 0A1 PIP could be used too.
When non-repudiation of receipt is required	Often there are several steps in a non-repudiation mechanism (or layers). Validation of the payload may not belong to the initial step. Also, it appears that different users give different meaning to non-repudiation, e.g. regarding the degree of payload validation. For these reasons, two options are available to users, depending on the precise semantics of non-repudiation that is required.
	Option 1: No hash (digest) is required in the receipt, and no payload validation is required. This is just a certification that the ebMS message has been well received. In such case, the ebMS 2.0 signed acknowledgement feature should be used. Only a reference to the MessageId of the acknowledged message is included in the Acknowledgement message. The *ackSignatureRequested* element of the CPA must be set.
	Option 2: A hash (digest) is required in the receipt. This option may also assume payload validation. In that case, an RN Receipt Acknowledgement with digest will be sent back. From an ebMS messaging viewpoint, this is a business message to be handled in the same way as any other action message.

7 TPA Features Specific to QoS and Deployment Configurations

7.1 Security

Security will follow the ebMS 2.0 specification requirements.

- Signature of the ebMS header as well as ebXML payload(s) is supported by XMLDSIG.
- Confidentiality of the ebXML Headers is supported by S/MIME (XML Encryption was not finalized at the time ebMS2 has been released).
- Confidentiality of the ebXML payloads (SOAP attachments) can be supported by S/MIME, or another method (XML Encryption may be used). This has to be specified in the TPA (and CPA)

7.2 Reliability

Using the Reliable messaging feature (Guaranteed Delivery and Duplicate Elimination) is recommended. In particular, Guaranteed Delivery will provide the following features:

- A notice of failure to the sender in case of non-delivery.
- A first level of Acknowledgement (not visible outside the MSH, used to control a message resending mechanism).
- A message resending mechanism, controlled by parameters specified in the CPA (some of them mapping to PIP definition parameters)
- Optionally, a signed acknowledgement that can be used as a first step in non-repudiation. However, this acknowledgement does not contain a digest of the original message.

7.3 Occasionally Connected Partners

In ebMS 2.0, a party has no other way to figure that a partner's MSH is down other than by getting repeated delivery failure notices, when trying to send messages to this partner. Dealing with occasionally connected partners may be done in two ways, that both involve a particular behavior from the layer above messaging:

(a) In case the periods of non-connectivity are known, both partners should share such information using a TPA structure above. A sender would then avoid sending messages to the non-connected party while it is down.
(b) In case messages are still sent to a non-connected partner, the reliable messaging feature may trigger useless message resending, but in all cases will notify of delivery failures on sender side. A series of failure delivery notices should be interpreted by the application layer as a sign that efforts to send messages to this party should stop, until the connectivity can be verified in other ways.

7.4 Routing and Multi-Hop

We consider here a two-hop scenario, via a Hub connection. The basic invariant of multi-hop, is that the ebMS message is not altered along the routing path. This is different from a routing mode where the Hub is treated as a final destination, and forwards the message payload based within a new ebMS envelope, e.g. with a new eb:To/PartyId, possibly with different QoS requirements. This mode is certainly possible but out of scope for this profiling specification.

We assume here that the final destination is already mentioned in the original message (eb:To / PartyID). We can roughly classify the different ways to achieve multi-hop routing as follows:

1. Transport-level routing: The Hub is not acting as an ebXML MSH, and not even as a SOAP intermediary. It uses a routing mechanism that is independent from the ebMS header content. In that case the routing is transparent to the ebXML function. Such mechanism could be based on URL attributes.

2. The Hub is acting as a SOAP intermediary. In that case also the routing is transparent to the ebXML function. Such mechanism could be based on some extra SOAP Header block (other than ebMS blocks), such as WS-Addressing header wsa:To. In such a case, SOAP faults may be generated.

3. The Hub is acting as an ebXML MSH – i.e. it has a routing mechanism that is dependent on the ebMS header content. Typically, the To/PartyId header element would be used to determine the routing. The Hub will act in the "NextMSH" role, while the final MSH destination will act in both the "ToPartyMSH" and "NextMSH" roles. The Hub may then consume the *MessageHeader*/*To* header element (that must not be set to *ToPartyMSH* actor, in that case). In case the routing function requires other data, it MAY use data in the message body provided that this data does not conflict with ebMS header data.

    a. Regarding the reliability feature: no acknowledgement from the Hub (no intermediary Acks) should be expected. This profile requires that the actor attribute of the AckRequested element be set to *ToPartyMSH*.
    b. Regarding Security: header elements used for routing must not encrypted. The Hub is not supposed to have security capabilities.

.

8 Miscellaneous

None.

9 Appendix A: CPA Profiling and Sample

9.1 CPA Profiling Forms

The profiling and definition of CPA data (both instance and profile template) can be facilitated using a set of forms, such as those provided by the ebXML Implementation, Interoperability and Conformance (IIC) OASIS Committee. It is strongly recommended for business partners to use the "Deployment Profile Template for CPPA V2.0" published by the OASIS IIC, in order to finalize their collaboration agreements. A subset of these forms is presented here.
Each element (or entry) in each one of these forms maps to a CPA element. Either the name of the entry is explicit enough to refer to the corresponding CPA element, or the name of the corresponding CPA element is mentioned in clear, usually prefixed with the qualifier "tp:" (e.g. tp:channelID).

- When entries in these forms must map to some PIP definition elements, it is indicated in the form entry.
- When entries in these forms are left to the user to instantiate as s/he wants to, the entry value is left empty (or just referring to the actual name of the CPA element, e.g. tp:TransportID)

A sample CPA document is listed in the next sub-section.

NOTE: These forms and their content are based on CPPA V2.1, which is very close to V2.0 (includes an errata from V2.0 and has additional extensibility points - some element names may be different. Please refer to the Errata for V2.0.)

9.2 Profiling the CPA Artifact Names and References

This form is used to identify the CPA profile, and also any CPA instance that is derived from a profile. It recommends some naming conventions for the CPA artifacts.

CPA Profile Info		
CPA Profile Info	**Name**	[Provide a name for the Collaboration Protocol Agreement profile. The name should identify when applicable: (a) the version of CPA, (b) the community sharing this profile (here, RN), (c) type of artifact (here a profile), (d) name of profile, (e) party ID if this profile is attached to a party.] Recommended: "CPA2.0-RN-Profile-"<profileID>"-"<partner1>" Examples: CPA2.0-RN-Profile-PIP3A4-222222 CPA2.0-RN-Profile-TP31-222222

	File name	[Provide a file name for the Collaboration Protocol Agreement profile file.] "CPA2.0-RN-Profile-"<profileID>"-"<partner1>"-file" (followed by appropriate suffix – e.g. .xml for the XML definition.) Examples: CPA2.0-RN-Profile-PIP3A4-222222-file.pdf CPA2.0-RN-Profile-TP31-222222-file.xml
CPA Instance Info	**Name**	[Define the name format for the CPA instances resulting from using this profile. The name should identify when applicable: (a) the version of CPA, (b) the community sharing this profile, (c) name of profile, (d) ID of instance, (e) party IDs.] Recommended: "CPA2.0-RN-"<profileID>"-"<instID>"-"<partner1-partner2> Example: CPA2.0-RN -P15-001-222222-333333 CPA2.0-RN -TP2-004-222222-333333
	File name	[Define the file name format for a Collaboration Protocol Agreement instance.] Recommended: "CPA2.0-RN-"< profileID >"-"<instID>"-"<partner1-partner2>"-file" (followed by appropriate suffix – e.g. .xml for the XML definition.) Example: CPA2.0-RN-P15-001-222222-333333-file.pdf CPA2.0-RN-TP2-004-222222-333333-file.xml
	CPA Id	[Define the format of the CPA Id. Must align with CPAId in message header.] Recommended: same as CPA name, i.e.: "CPA2.0-RN-"< profileID >"-"<instID>"-"<partner1-partner2>
	Lifetime of CPA	**Start**: [The starting date and time of the agreement.]
		End: [The end date and time of the agreement. The start and end date/times define the duration that the agreement is in effect.]
	Context of application	**ConversationLimit**: [NONE or numeric value. The agreement is terminated (no longer valid) when the conversation limit is reached.]
		Concurrent Conversation Limit: [NONE or numeric value. The maximum number of conversations that can be in process at the same time. Provide this value when there are constraints that limit the number of business transactions that one or more of the parties can process simultaneously.]

9.3 Profiling the Party Info

This form is used to identify the parties involve. A CPA profile will typically contain one of these fully instantiated. At least another one of these will need to be filled by another business partner in order to produce a complete CPA instance.

Profiling (alignment with data or QoS in Rosettanet PIPs, or with ebMS header data that is itself profiled) is required for some entries of this table The rest of this table is provided as a support for users.

Party Info		
CPA Reference	[CPA Profile name]	
	[CPA Instance name, if used for instantiating a particular CPA]	
Party element	**PartyId**	[The formal unique identifier for the organization. Must align with eb:PartyId in message header (section 4)] All Party ID elements present in CPA must appear in the message header**.**
	Type	[Must align with eb:PartyId/@type in message header (section 4)]
	Reference	[A URL or URI that points to a location (e.g. web page or directory) where more information can be found on the party.]
Collaboration Roles elements	[List the collaboration role names that this party is expected to fulfill. The role names need to be unique within this list. Each role will be detailed in a CollaborationRole form.]	
	CollaborationRole 1	Process Name [maps to eb:Service I header] Role Name [maps to eb:Role in header]
	CollaborationRole 2	Process Name [maps to eb:Service I header] Role Name [maps to eb:Role in header]
	(others?)	
Certificates elements	[List the certificates info and ID.]	
	Certificate 1	
	Certificate 2	
	(others?)	
DeliveryChannels elements	[describes a *Party's Message*-receiving and *Message*-sending characteristics. It consists of one document-exchange definition and one transport definition. The details of each DeliveryChannel element will be specified in a different form.]	
	DeliveryChannel 1	[give only the tp:channelId]
	DeliveryChannel 2	[give only the tp:channelId]
	(others?)	

Transports elements		
	Transport ID	[tp:TransportId]
Documents Exchanges		
	Exchange ID	[tp:docExchangeId]

9.4 Profiling the Collaboration Roles

This form is used to identify the roles in which a party may be acting under this CPA or CPA profile. One form will be filled for each role.

Profiling (alignment with data or QoS in Rosettanet PIPs) is required for some entries of this table The rest of this table is provided as a support for users.

CollaborationRole Info		
CPA Reference	[CPA Profile name]	
	[CPA Instance name, if used for instantiating a particular CPA]	
Role Identification	Name	[maps to eb:Role]
	Type	[xlink:type], e.g. “simple”
	href	[xlink:href] Example: xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer">
Application Certificate	ID:	
	Comments:	
Process Specification	name	[The name of the business process specification that this role applies to.] maps to ProcessSpecification nameID attribute in BPSS guideline (e.g. **urn:rosettanet:specification:interchange:PIP3A4:xml:bpss:d11_00), i.e. to eb:Service (see Section 4 Message Description)** xlink:href : contains a reference to the BPSS definition (e.g. =http://www.rosettanet.org/processes/3A4.xml)
	version	[Version of the business process specification]
	type	
	Uuid / nameId	tradingpartner uuid → attribute uuid of BPSS definition when present (attr in process specification top element) (Example= "urn:icann:rosettanet.org:bpid:3A4$2.0")
Service Binding item (One for every Action	Associated Service name	[tp:ServiceBinding/tp:Service value] Maps to eb:Service **(see Section 4 Message Description)** Example: <tp:Service>**urn:rosettanet:specification:interchange:PIP3A4:xml:b**

or Signal message)		**pss:d11_00**</tp:Service>	
	Action direction	[send / receive]	
	Action Binding	[tp:id] example: companyA_ABID1 (to be used for further references. Unique)	
		[tp:action] example: "Purchase Order Request Action" maps to eb:Action (**see Section 4, Message Description**)(e.g. ="PurchaseOrderRequestAction")	
		[tp:packageId] Exammmple: tp:packageId="CompanyA_RequestPackage". Refers to MIME structure of payload.	
	Business Transaction Characteristics	tp:isNonRepudiationRequired	maps to “Non-Repudiation of Origin and Content”, column 8 in PIP definition tables below. (="true" in below example)
		tp:isNonRepudiationReceiptRequired	maps to “Non-Repudiation Required” column 3 in PIP tables below. (="true" in below example)
		tp:isConfidential	(using SSL or digital envelope)
		tp:isAuthenticated	NOTE: should map to DTD related docs
		tp:isTamperProof	(NOTE: authenticated gives integrity)
		tp:isAuthorizationRequired	maps to “Is Authorization Required“ column 7 in PIP tables below. (="true" in below example)
		tp:timeToAcknowledgeReceipt	maps to “Time to Acknowledge“ column 4 in PIP tables below. (="PT2H" in below example) NOTE: it should be equivalent to (retryInterval * Retries) in ebMS.
		tp:timeToPerform	maps to “Time to Perform“ column 5 in PIP tables below (not really captured in CPA, about same as time to Ack)
		Tp: isIntelligibleCheckRequired	
		Tp: timeToAcknowledgeReceipt	
		Tp: timeToAcknowledgeAcceptance	
		Tp: ***retryCount***	Must NOT be used. Instead, the Retries element of the ReliableMessaging CPA element will map to “Retry Count“ column 6 in PIP tables below.

9.5 Profiling the Delivery Channels

Delivery Channels - A delivery channel describes a *Party's Message*-receiving and *Message*-sending characteristics. It consists of one document-exchange definition and one transport definition.

No profiling is required for this data. This table is provided as a support for users.

Delivery Channel Info		
CPA Reference	[CPA Profile name]	
	[CPA Instance name, if used for instantiating a particular CPA]	
Identity and Components	**channelId**	
	transportId	
	docExchangeId	
Messaging Characteristics		
	syncReplyMode	
	ackRequested	Reliable Messaging parameter for Guaranteed Delivery (At Least Once)
	ackSignatureRequested	NOTE: this is a way to support a form of non-repudiation of Receipt, that is generally not sufficient for RosettaNet.
	duplicateElimination	Reliable Messaging parameter for No Duplicate Delivery (At Most Once)
	actor	

9.6 Profiling the Document Exchanges

Document Exchange - The Document-exchange layer specifies processing of the business documents by the Message-exchange function. Properties specified include encryption, digital signature, and reliable-messaging characteristics. The options selected for the Document-exchange layer are complementary to those selected for the transport layer. For example, if Message security is desired and the selected transport protocol does not provide *Message* encryption, then *Message* encryption must be specified in the Document-exchange layer.

Profiling (alignment with data or QoS in Rosettanet PIPs) is required for some entries of this table The rest of this table is provided as a support for users.

Document Exchange Info		
CPA Reference	[CPA Template name]	
	[CPA Instance name, if used for instantiating a particular CPA]	
Doc Exchange ID	[tp:docExchangeId]	
Sender Binding	**Reliable Messaging**	**[**tp:ReliableMessaging] - tp:Retries: [maps to "Retry Count" column 6 in above tables.] - tp:RetryInterval: [Example: <tp:RetryInterval>PT2H</tp:RetryInterval>] - tp:MessageOrderSemantics: [Example: "Guaranteed"]
	Persist Duration	**[**tp:PersistDuration]
	Non Repudiation of Origin	[tp:SenderNonRepudiation] - tp:NonRepudiationProtocol - tp:HashFunction - tp:SignatureAlgorithm - tp:SigningCertificateRef
	Digital Envelope	**[**tp:SenderDigitalEnvelope] - tp:DigitalEnvelopeProtocol - tp:EncryptionAlgorithm - tp:EncryptionSecurityDetailsRef
	Nemespaces	**[**tp:NamespaceSupported]
Receiver Binding		
	Reliable Messaging	**[**tp:ReliableMessaging] - tp:Retries - tp:RetryInterval - tp:MessageOrderSemantics
	Persist Duration	**[**tp:PersistDuration]
	Non Repudiation of Receipt	[tp:ReceiverNonRepudiation] - tp:NonRepudiationProtocol - tp:HashFunction - tp:SignatureAlgorithm - tp:SigningSecurityDetailsRef
	Digital Envelope	**[**tp:ReceiverDigitalEnvelope] - tp:DigitalEnvelopeProtocol - tp:EncryptionAlgorithm - tp:EncryptionCertificateRef
	Namespaces	**[**tp:NamespaceSupported]

9.7 Profiling the Transport Protocol

The transport layer identifies the transport protocol to be used in sending messages through the network and defines the endpoint addresses, along with various other properties of the transport protocol. Choices of properties in the transport layer are complementary to those in the document-exchange layer (see "Document-Exchange Layer" directly above.)

No profiling is required for this data. This table is provided as a support for users.

Transport Info		
CPA Reference	[CPA Template name]	
	[CPA Instance name, if used for instantiating a particular CPA]	
Transport Sender	**protocol**	**[tp:** TransportProtocol]
	Client security	**[**tp:TransportSecurityProtocol]
		[tp:ClientCertificateRef]
Transport Receiver	**protocol**	**[tp:** TransportProtocol]
	End Point	[tp:Endpoint/@uri, tp:Endpoint/@type]
	Server security	**[**tp:TransportSecurityProtocol]
		[tp:ServerCertificateRef]
		[tp:ClientSecurityDetailsRef]

9.8 Examples of Tables Used in PIP Definitions

These tables are extracted from the PIP7C7 definition. Their purpose here is to illustrate the terms and properties that map to the concepts in above CPA forms. The last row in these tables has been added to identify columns that are referred to in the above CPA forms.

Table 7: Business Activity Performance Controls

Role Name	Activity Name	Acknowledgment of Receipt			Retry Count	Is Authorization Required?	Non-Repudiation of Origin and Content?
		Non-Repudiation Required?	Time to Acknowledge	Time to Perform			
Foundry or Test Services	Notify of Semiconductor Test Data	Y	2 hrs	N/A	3	Y	Y

Table 10: Message Exchange Controls

#	Name	Time to Acknowledge	Time to Respond to Action	Included in Time to Perform	Is Authorization Required?	Is Non-Repudiation Required?	Is Secure Transport Required?
1.	Semiconductor Test Data Notification Action	2 hrs	N/A	N/A	Y	Y	Y
1.1.	Receipt Acknowledgment	N/A	N/A	N/A	N	N	Y

9.9 Sample CPA Material

This is extracted from a sample CPA V2.0 document. Note: this is not a complete CPA document.

<... tp:partyName="CompanyA">

<tp:PartyId>

tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns"

123456789

</tp:PartyId>

<tp:PartyRef xlink:href="http://CompanyA.com/about.html"/>

<tp:ProcessSpecification>
tp:version="2.0"
tp:name="PIP3A4RequestPurchaseOrder"
xlink:href="http://www.rosettanet.org/processes/3A4.xml"
tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
</tp:ProcessSpecification>

<tp:Role>
tp:name="Buyer"
xlink:type="simple"
xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer">
</tp:Role>

<tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>

Security Certificate References and Certificates

tp:action="Purchase Order Request Action"

tp:packageId="CompanyA_RequestPackage"

References the detailed information describing the way the data is assembled before entering the network.

<tp:BusinessTransactionCharacteristics
tp:isNonRepudiationRequired="true"
tp:isNonRepudiationReceiptRequired="true"
tp:isConfidential="transient"
tp:isAuthenticated="persistent"
tp:isTamperProof="persistent"
tp:isAuthorizationRequired="true"
tp:timeToAcknowledgeReceipt="PT2H"
tp:timeToPerform="P1D"/>


```
<tp:ActionContext
        tp:binaryCollaboration="Request Purchase Order"
        tp:businessTransactionActivity="Request Purchase Order"
        tp:requestOrResponseAction="Purchase Order Request Action"/>
```

From the BPSS instance

```
<tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
      <tp:AccessAuthentication>basic</tp:AccessAuthentication>
      <tp:AccessAuthentication>digest</tp:AccessAuthentication>
      <tp:Endpoint
       tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
       tp:type="allPurpose"/>
      <tp:TransportServerSecurity>
       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
       <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
       <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>

<tp:ReliableMessaging>
        <tp:Retries>3</tp:Retries>
        <tp:RetryInterval>PT2H</tp:RetryInterval>
        <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
      </tp:ReliableMessaging>

<tp:SenderNonRepudiation>

<tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
        <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
        <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
sha1</tp:SignatureAlgorithm>
        <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
      </tp:SenderNonRepudiation>

<tp:SenderDigitalEnvelope>
        <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
        <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
        <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
      </tp:SenderDigitalEnvelope>

<tp:PersistDuration>P1D</tp:PersistDuration>
```

10 Appendix B: Glossary

AMD	Abstract Message Service
ATPA	Abstract Trading Partner Agreement
MEP	Message Exchange Profiles
RNIF	RosettaNet™ Implementation Framework
PIP	(RosettaNet terminology): Partner Interface Process
TP	Trading Profile
ebMS	ebXML Messaging Services specification (an ebXML standard)
BPSS	ebXML Business Process Specification Schema (an ebXML standard)
ebBP	ebXML Business Process specification (applies to new version of BPSS, renamed)
CPP	ebXML Collaboration Protocole Profile (desribed in CPPA specification, an ebXML standard)
CPA	ebXML Collaboration Protocole Agreement (desribed in CPPA specification, an ebXML standard)
SBDH	Standard Business Document Header (also known as “Generic Header”)

11 References

[AMD] *MMS Abstract Message Definition*, Draft 00.07.00, 7 January 2005

[ebMS] OASIS, *ebXML Message Service Specification Version 2.0*, http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf, April 1, 2002.

[DPT-ebMS2] *Deployment Profile Template 1.1 for ebMS 2.0*, OASIS IIC committee draft, July 2005.
http://www.oasis-open.org/committees/documents.php?wg_abbrev=ebxml-iic

[DPT-CPPA2] *Deployment Profile Template 0.2 for CPPA 2.0*, OASIS IIC working draft, August 2005.
http://www.oasis-open.org/committees/documents.php?wg_abbrev=ebxml-iic

[**BPSS-PIP**] *RosettaNet ebXML BPSS Guideline*, v1.11, August 2004.

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[RN-NameSpaces] *RosettaNet Namespace Specification and Management*, v1.0, December 2003.

