TECHNICAL SPECIFICATION

ISO/TS 15000-1

First edition
2004-05-15

Electronic business eXtensible Markup Language (ebXML) —

Part 1: Collaboration-protocol profile and agreement specification (ebCPP)

Commerce électronique en langage de balisage extensible (ebXML) —

Partie 1: Spécification de l'accord et du profil de protocole de collaboration (ebCPP)

Reference number
ISO/TS 15000-1:2004(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

In other circumstances, particularly when there is an urgent market requirement for such documents, a technical committee may decide to publish other types of normative document:

— an ISO Publicly Available Specification (ISO/PAS) represents an agreement between technical experts in an ISO working group and is accepted for publication if it is approved by more than 50 % of the members of the parent committee casting a vote;

— an ISO Technical Specification (ISO/TS) represents an agreement between the members of a technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/PAS or ISO/TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/PAS or ISO/TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/TS 15000-1 was prepared by OASIS ebXML Collaboration Protocol Profile and Agreement Technical Committee (as Collaboration-Protocol Profile and Agreement Specification Version 2.0) and was adopted by Technical Committee ISO/TC 154, *Processes, data elements and documents in commerce, industry and administration*. The content of ISO/TS 15000-1 and Collaboration-Protocol Profile and Agreement Specification Version 2.0 is identical.

ISO/TS 15000 consists of the following parts, under the general title *Electronic business eXtensible Markup Language (ebXML)*:

— *Part 1: Collaboration-protocol profile and agreement specification (ebCPP)*

— *Part 2: Message service specification (ebMS)*

— *Part 3: Registry information model specification (ebRIM)*

— *Part 4: Registry services specification (ebRS)*

OASIS/ebXML Collaboration-Protocol Profile and Agreement Specification v2.0

Approved OASIS Standard

OASIS/ebXML Collaboration Protocol Profile and Agreement Technical Committee

September 2002

1 Status of this Document

This document specifies an ebXML SPECIFICATION for the eBusiness community.

Distribution of this document is unlimited.

The document formatting is based on the Internet Society's Standard RFC format.

This version:

http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0.pdf

Errata for this version:

http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0-Errata.shtml

***Previous version*:**

http://www.ebxml.org/specs/ebCCP.pdf

2 Technical Committee Members

Selim Aissi	Intel
Arvola Chan	TIBCO
James Bryce Clark	Individual Member
David Fischer	Drummond Group
Tony Fletcher	Individual Member
Brian Hayes	Collaborative Domain
Neelakantan Kartha	Sterling Commerce
Kevin Liu	SAP
Pallavi Malu	Intel
Dale Moberg	Cyclone Commerce
Himagiri Mukkamala	Sybase
Peter Ogden	Cyclone Commerce
Marty Sachs	IBM
Yukinori Saito	Individual Member
David Smiley	Mercator
Tony Weida	Individual Member
Pete Wenzel	SeeBeyond
Jean Zheng	Vitria

3 ebXML Participants

The authors wish to recognize the following for their significant participation in developing the Collaboration Protocol Profile and Agreement Specification, Version 1.0.

David Burdett, CommerceOne
Tim Chiou, United World Chinese Commercial Bank
Chris Ferris, Sun
Scott Hinkelman, IBM
Maryann Hondo, IBM
Sam Hunting, ECOM XML
John Ibbotson, IBM
Kenji Itoh, JASTPRO
Ravi Kacker, eXcelon Corp.
Thomas Limanek, iPlanet
Daniel Ling, VCHEQ
Henry Lowe, OMG
Dale Moberg, Cyclone Commerce
Duane Nickull, XML Global Technologies
Stefano Pogliani, Sun
Rebecca Reed, Mercator
Karsten Riemer, Sun
Marty Sachs, IBM
Yukinori Saito, ECOM
Tony Weida, Edifecs

4 Table of Contents

5 Introduction

5.1 Summary of Contents of Document

As defined in the ebXML Business Process Specification Schema[ebBPSS], a *Business Partner* is an entity that engages in *Business Transactions* with another *Business Partner(s)*. The *Message*-exchange capabilities of a *Party* MAY be described by a *Collaboration-Protocol Profile (CPP)*. The *Message*-exchange agreement between two *Parties* MAY be described by a *Collaboration-Protocol Agreement (CPA)*. A *CPA* MAY be created by computing the intersection of the two *Partners*' *CPPs*. Included in the *CPP* and *CPA* are details of transport, messaging, security constraints, and bindings to a *Business-Process-Specification* (or, for short, *Process-Specification*) document that contains the definition of the interactions between the two *Parties* while engaging in a specified electronic *Business Collaboration*.

This specification contains the detailed definitions of the *Collaboration-Protocol Profile (CPP)* and the *Collaboration-Protocol Agreement* (*CPA)*.

This specification is a component of the suite of ebXML specifications.

The rest of this specification is organized as follows:

- Section 6 defines the objectives of this specification.
- Section 7 provides a system overview.
- Section 8 contains the definition of the *CPP*, identifying the structure and all necessary fields.
- Section 9 contains the definition of the *CPA*.
- Section 10 lists all other documents referenced in this specification.
- Section 11 provides a conformance statement.
- Section 12 contains a disclaimer.
- Section 13 lists contact information for the contributing authors and the coordinating editor for this version of the specification.
- The appendices include examples of *CPP* and *CPA* documents (non-normative), an example XML *Business Process Specification* (non-normative), an XML Schema document (normative), a description of how to compose a *CPA* from two *CPPs* (non-normative), a summary of corresponding ebXML Messaging Service and *CPA* parameters (normative), and a Glossary of Terms.
-

5.2 Document Conventions

Terms in *Italics* are defined in Appendix G (Glossary of Terms). Terms listed in ***Bold Italics*** represent the element and/or attribute content of the XML *CPP, CPA*, or related definitions.

In this specification, indented paragraphs beginning with "NOTE:" provide non-normative explanations or suggestions that are not mandated by the specification.

References to external documents are represented with BLOCK text enclosed in brackets, e.g. [RFC2396]. The references are listed in Section 10, "References".

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC 2119].

> NOTE: Vendors SHOULD carefully consider support of elements with cardinalities (0 or 1) or (0 or more). Support of such an element means that the element is processed appropriately for its defined function and not just recognized and ignored. A given *Party* might use these elements in some *CPPs* or *CPAs* and not in others. Some of these elements define parameters or operating modes and SHOULD be implemented by all vendors. It might be appropriate to implement elective elements that represent major run-time functions, such as various alternative communication protocols or security functions, by means of plug-ins so that a given *Party* MAY acquire only the needed functions rather than having to install all of them.

By convention, values of [XML] attributes are generally enclosed in quotation marks, however those quotation marks are not part of the values themselves.

5.3 Versioning of the Specification and Schema

Whenever this specification is modified, it SHALL be given a new version number.

It is anticipated that during the review period, errors and inconsistencies in the specification and in the schema may be detected and have to be corrected. All known errors in the specification as well as necessary changes to the schema will be summarized in an errata page found at

http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0-Errata.shtml

The specification, when initially approved by the OASIS ebXML Collaboration Protocol Profile and Agreement Technical Committee for public review, SHALL carry a version number of "2_0". At that time, the schema SHALL have a version number of "2_0b" and the suffix letter after "2_0" will be advanced as necessary when bug fixes to the schema have to be introduced. Such versions of the schema SHALL be found under the directory

http://www.oasis-open.org/committees/ebxml-cppa/schema/

In addition, the latest version of the schema SHALL always be found at

http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd

since the latter is the namespace URI used for this specification and the corresponding schema is supposed to be directly resolvable from the namespace URI.

The value of the version attribute of the Schema element in a given version of the schema SHALL be equal to the version of the schema.

5.4 Definitions

Technical terms in this specification are defined in Appendix G.

5.5 Audience

One target audience for this specification is implementers of ebXML services and other designers and developers of middleware and application software that is to be used for conducting electronic *Business*. Another target audience is the people in each enterprise who are responsible for creating *CPPs* and *CPAs*.

5.6 Assumptions

It is expected that the reader has an understanding of XML and is familiar with the concepts of electronic *Business* (eBusiness).

5.7 Related Documents

Related documents include ebXML Specifications on the following topics:

- ebXML *Message* Service Specification[ebMS]
- ebXML Business Process Specification Schema[ebBPSS]
- ebXML Core Component Overview[ccOVER]
- ebXML Registry Services Specification[ebRS]

See Section 10 for the complete list of references.

6 Design Objectives

The objective of this specification is to ensure interoperability between two *Parties* even though they MAY procure application software and run-time support software from different vendors. The *CPP* defines a *Party's Message*-exchange capabilities and the *Business Collaborations* that it supports. The *CPA* defines the way two *Parties* will interact in performing the chosen *Business Collaboration*s. Both *Parties* SHALL use identical copies of the *CPA* to configure their run-time systems. This assures that they are compatibly configured to exchange *Messages* whether or not they have obtained their run-time systems from the same vendor. The configuration process MAY be automated by means of a suitable tool that reads the *CPA* and performs the configuration process.

In addition to supporting direct interaction between two *Parties,* this specification MAY also be used to support interaction between two *Parties* through an intermediary such as a portal or broker.

It is an objective of this specification that a *CPA* SHALL be capable of being composed by intersecting the respective *CPPs* of the *Parties* involved. The resulting *CPA* SHALL contain only those elements that are in common, or compatible, between the two *Parties*. Variable quantities, such as number of retries of errors, are then negotiated between the two *Parties.* The design of the *CPP* and *CPA* schemata facilitates this composition/negotiation process. However, the composition and negotiation processes themselves are outside the scope of this specification. Appendix E contains a non-normative discussion of this subject.

It is a further objective of this specification to facilitate migration of both traditional EDI-based applications and other legacy applications to platforms based on the ebXML specifications. In particular, the *CPP* and *CPA* are components of the migration of applications based on the X12 838 Trading-Partner Profile[X12] to more automated means of setting up *Business* relationships and doing *Business* under them.

7 System Overview

7.1 What This Specification Does

The exchange of information between two *Parties* requires each *Party* to know the other *Party's* supported *Business Collaborations*, the other *Party's* role in the *Business Collaboration,* and the technology details about how the other *Party* sends and receives *Messages*. In some cases, it is necessary for the two *Parties* to reach agreement on some of the details.

The way each *Party* can exchange information, in the context of a *Business Collaboration*, can be described by a *Collaboration-Protocol Profile (CPP).* The agreement between the *Parties* can be expressed as *a Collaboration-Protocol Agreement (CPA).*

A *Party* MAY describe itself in a single *CPP*. A *Party* MAY create multiple *CPPs* that describe, for example, different *Business Collaborations* that it supports, its operations in different regions of the world, or different parts of its organization.

To enable *Parties* wishing to do *Business* to find other *Parties* that are suitable *Business Partners*, *CPP*s MAY be stored in a repository such as is provided by the ebXML Registry[ebRS]. Using a discovery process provided as part of the specifications of a repository, a *Party* MAY then use the facilities of the repository to find *Business Partners*.

The document that defines the interactions between two *Parties* is a *Process-Specification* document that MAY conform to the ebXML Business Process Specification Schema[ebBPSS]. The *CPP* and *CPA* include references to this *Process-Specification* document. The *Process-Specification* document MAY be stored in a repository such as the ebXML Registry. See NOTE about alternative *Business-Collaboration* descriptions in Section 8.4.4.

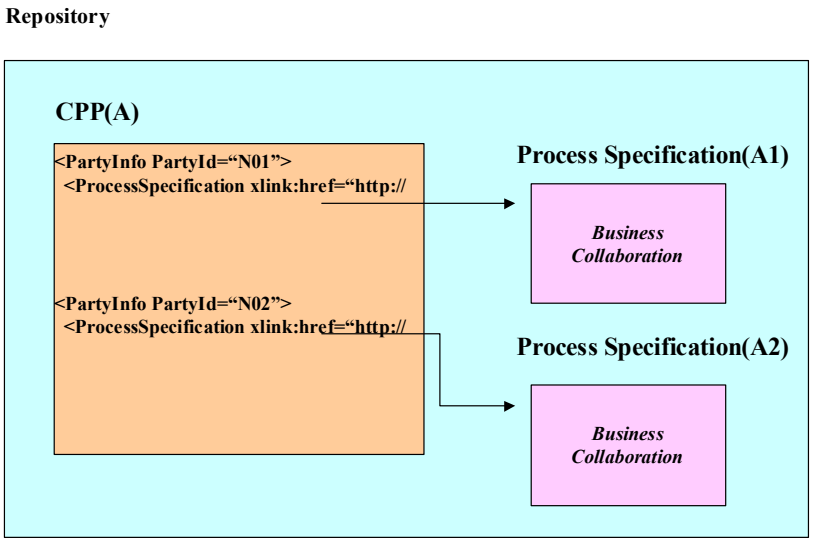
Figure 1 illustrates the relationships between a *CPP* and two *Process-Specification* documents, A1 and A2, in an ebXML Registry. On the left is a *CPP*, A, which includes information about two parts of an enterprise that are represented as different *Parties*. On the right are shown two *Process-Specification* documents. Each of the ***PartyInfo*** elements in the *CPP* contains a reference to one of the *Process-Specification* documents. This identifies the *Business Collaboration*s that the *Party* can perform.

This specification defines the markup language vocabulary for creating electronic *CPPs* and *CPAs*. *CPPs* and *CPAs* are [XML] documents. In the appendices of this specification are two sample *CPPs*, a sample *CPA* formed from the *CPPs*, a sample *Process-Specification* referenced by the *CPPs* and the *CPA*, and the XML Schema governing the structures of *CPPs* and *CPAs*.

The *CPP* describes the capabilities of an individual *Party*. A *CPA* describes the capabilities that two *Parties* have agreed to use to perform particular *Business Collaboration*s. These *CPAs* define the "information technology terms and conditions" that enable *Business* documents to be electronically interchanged between *Parties.* The information content of a *CPA* is similar to the information-technology specifications sometimes included in Electronic Data Interchange (EDI)

390 *Trading Partner Agreements (TPAs).* However, these *CPAs* are not paper documents. Rather,
391 they are electronic documents that can be processed by computers at the *Parties'* sites in order to
392 set up and then execute the desired *Business* information exchanges. The "legal" terms and
393 conditions of a *Business* agreement are outside the scope of this specification and therefore are
394 not included in the *CPP* and *CPA*.

Figure 1: Structure of CPP & Business Process Specification in an ebXML Registry



395
396 An enterprise MAY choose to represent itself as multiple *Parties*. For example, it might
397 represent a central office supply procurement organization and a manufacturing supplies
398 procurement organization as separate *Parties.* The enterprise MAY then construct a *CPP* that
399 includes all of its units that are represented as separate *Parties.* In the *CPP*, each of those units
400 would be represented by a separate ***PartyInfo*** element.
401
402 The CPPA specification is concerned with software that conducts business on behalf of *Parties*
403 by exchanging *Messages*[ebMS]. In particular, it is concerned with *Client* and *Server* software
404 programs that engage in *Business Transactions*[ebBPSS] by sending and receiving *Messages.*
405 Those *Messages* convey *Business Documents* and/or business signals[ebBPSS] in their payload.
406 Under the terms of a CPA:
407
408 • A *Client* initiates a connection with a *Server.*
409 • A *Requester* initiates a *Business Transaction* with a *Responder*.
410 • A *Sender* sends a *Message* to a *Receiver.*
411
412 Thus, *Client* and *Server* are software counterparts, *Requester* and *Responder* are business
413 counterparts, and *Sender* and *Receiver* are messaging counterparts. There is no fixed

relationship between counterparts of different types. For example, consider a purchasing collaboration. *Client* software representing the buying party might connect to *Server* software representing the selling party, and then make a purchase request by sending a *Message* containing a purchase order over that connection. If the CPA specifies a synchronous business response, the *Server* might then respond by sending a *Message* containing an acceptance notice back to the *Client* over the same connection. Alternatively, if the CPA specifies an asynchronous business response, *Client* software representing the selling party might later respond by connecting to *Server* software representing the buying party and then sending a *Message* containing an acceptance notice.

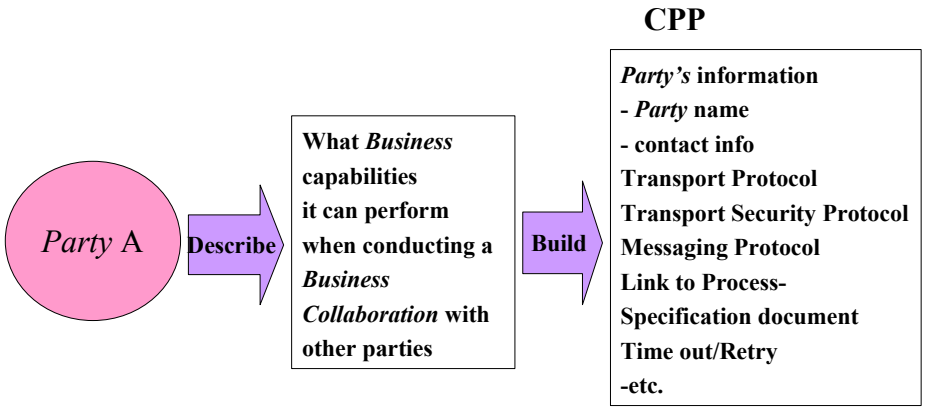
In general, the *Parties* to a *CPA* can have both client and server characteristics. A client requests services and a server provides services to the *Party* requesting services. In some applications, one *Party* only requests services and one *Party* only provides services. These applications have some resemblance to traditional client-server applications. In other applications, each *Party* MAY request services of the other. In that case, the relationship between the two *Parties* can be described as a peer-peer relationship rather than a client-server relationship.

7.2 Forming a CPA from Two CPPs

This section summarizes the process of discovering a *Party* to do *Business* with and forming a *CPA* from the two *Parties' CPP*s. In general, this section is an overview of a possible procedure and is not to be considered a normative specification. See Appendix E "CPA Composition (Non-Normative)" for more information.

Figure 2 illustrates forming a *CPP*. *Party* A tabulates the information to be placed in a repository for the discovery process, constructs a *CPP* that contains this information, and enters it into an ebXML Registry or similar repository along with additional information about the *Party*. The additional information might include a description of the *Businesses* that the *Party* engages in. Once *Party* A's information is in the repository, other *Parties* can discover *Party* A by using the repository's discovery services.

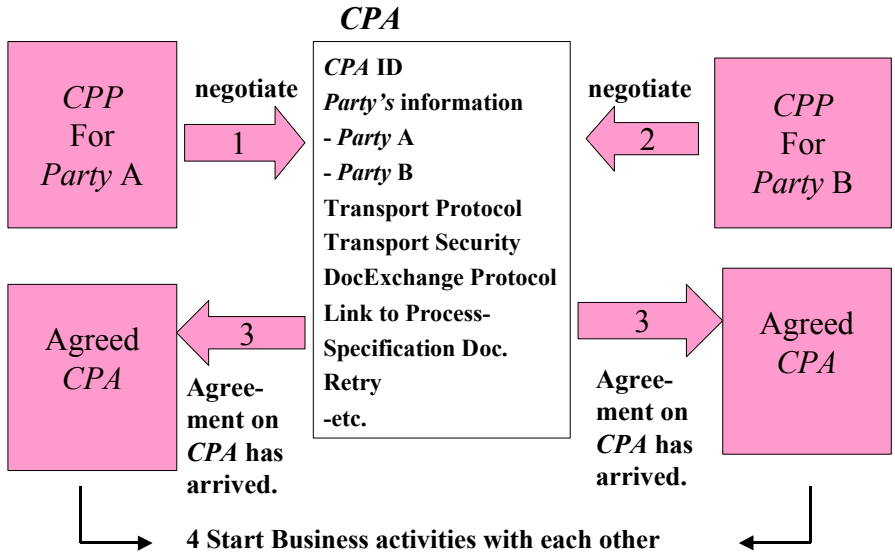
Figure 2: Overview of Collaboration-Protocol Profiles (CPP)



443

444 In Figure 3, *Party* A and *Party* B use their *CPP*s to jointly construct a single copy of a *CPA* by
445 calculating the intersection of the information in their *CPP*s. The resulting *CPA* defines how the
446 two *Parties* will behave in performing their *Business Collaboration*.

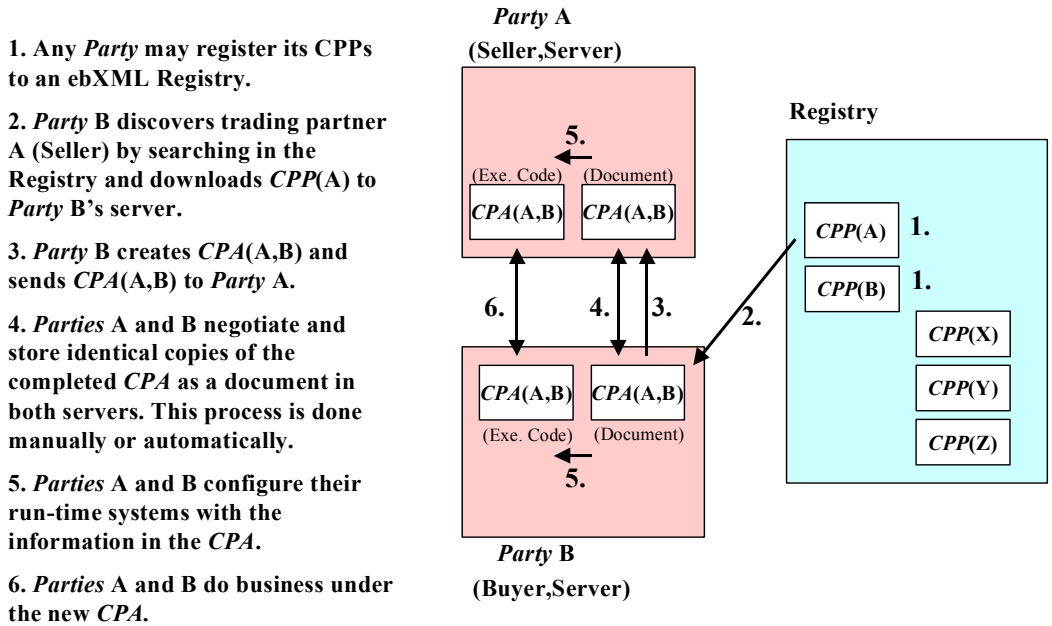
Figure 3: Overview of *Collaboration-Protocol Agreements* (*CPA*)



447

Figure 4 illustrates the entire process. The steps are listed at the left. The end of the process is that the two *Parties* configure their systems from identical copies of the agreed *CPA* and they are

Figure 4: Overview of Working Architecture of CPP/CPA with ebXML Registry



then ready to do *Business*.

7.3 Forming a CPA from a CPA Template

Alternatively, a *CPA* template might be used to create a *CPA*. A *CPA* template represents one party's "fill in the blanks" proposal to a prospective trading partner for implementing one or more business processes. For example, such a template might contain placeholder values for identifying aspects of the other party. To form a *CPA* from a *CPA* template, the placeholder values would be replaced by the actual values for the other trading partner. Actual values might be obtained from the other party's CPP, if available, or by data entry in an HTML form, among other possibilities. The current version of this specification does not address how placeholder values might be represented in a *CPA*. However, the process of filling out a *CPA* template MUST result in a valid *CPA*. Further discussion of *CPA* templates is provided in Appendix E.

7.4 How the CPA Works

A *CPA* describes all the valid visible, and hence enforceable, interactions between the *Parties* and the way these interactions are carried out. It is independent of the internal processes executed at each *Party*. Each *Party* executes its own internal processes and interfaces them with the *Business Collaboration* described by the *CPA* and *Process-Specification* document. The *CPA* does not expose details of a *Party's* internal processes to the other *Party.* The intent of the *CPA* is

to provide a high-level specification that can be easily comprehended by humans and yet is precise enough for enforcement by computers.

The information in the *CPA* is used to configure the *Parties'* systems to enable exchange of *Messages* in the course of performing the selected *Business Collaboration.* Typically, the software that performs the *Message* exchanges and otherwise supports the interactions between the *Parties* is middleware that can support any selected *Business Collaboration.* One component of this middleware MAY be the ebXML *Message* Service Handler[ebMS]. In this specification, the term "run-time system" or "run-time software" is used to denote such middleware.

The *CPA* and the *Process-Specification* document that it references define a conversation between the two *Parties*. The conversation represents a single unit of *Business* as defined by the ***BinaryCollaboration*** component of the *Process-Specification* document. The conversation consists of one or more *Business Transactions*, each of which is a request *Message* from one *Party* and zero or one response *Message* from the other *Party*. The *Process-Specification* document defines, among other things, the request and response *Message*s for each *Business Transaction* and the order in which the *Business Transactions* are REQUIRED to occur. See [ebBPSS] for a detailed explanation.

The *CPA* MAY actually reference more than one *Process-Specification* document. When a *CPA* references more than one *Process-Specification* document, each *Process-Specification* document defines a distinct type of conversation. Any one conversation involves only a single *Process-Specification* document.

A new conversation is started each time a new unit of *Business* is started. The *Business Collaboration* also determines when the conversation ends. From the viewpoint of a *CPA* between *Party* A and *Party* B, the conversation starts at *Party* A when *Party* A sends the first request *Message* to *Party* B. At *Party* B, the conversation starts when it receives the first request of the unit of *Business* from *Party* A. A conversation ends when the *Parties* have completed the unit of *Business*.

> NOTE: The run-time system SHOULD provide an interface by which the *Business* application can request initiation and ending of conversations.

7.5 Where the CPA May Be Implemented

Conceptually, a *Business*-to-*Business* (B2B) server at each *Party's* site implements the CPA and Process-Specification document. The B2B server includes the run-time software, i.e. the middleware that supports communication with the other *Party*, execution of the functions specified in the *CPA*, interfacing to each *Party's* back-end processes, and logging the interactions between the *Parties* for purposes such as audit and recovery. The middleware might support the concept of a long-running conversation as the embodiment of a single unit of *Business* between the *Parties*. To configure the two *Parties'* systems for *Business-to-Business* operations, the information in the copy of the *CPA* and *Process-Specification* documents at each *Party's* site is installed in the run-time system. The static information MAY be recorded in a local database and

other information in the *CPA* and *Process-Specification* document MAY be used in generating or customizing the necessary code to support the *CPA*.

> NOTE: It is possible to provide a graphical *CPP/CPA*-authoring tool that understands both the semantics of the *CPP/CPA* and the XML syntax. Equally important, the definitions in this specification make it feasible to automatically generate, at each *Party's* site, the code needed to execute the *CPA*, enforce its rules, and interface with the *Party's* back-end processes.

7.6 Definition and Scope

This specification defines and explains the contents of the *CPP* and *CPA* XML documents. Its scope is limited to these definitions. It does not define how to compose a *CPA* from two *CPPs* nor does it define anything related to run-time support for the *CPP* and *CPA*. It does include some non-normative suggestions and recommendations regarding *CPA* composition from two *CPPs* and run-time support where these notes serve to clarify the *CPP* and *CPA* definitions. See Section 11 for a discussion of conformance to this specification.

> NOTE: This specification is limited to defining the contents of the *CPP* and *CPA*, and it is possible to be conformant with it merely by producing a *CPP* or *CPA* document that conforms to the XML Schema document defined herein. It is, however, important to understand that the value of this specification lies in its enabling a run-time system that supports electronic commerce between two *Parties* under the guidance of the information in the *CPA*.

8 CPP Definition

A *CPP* defines the capabilities of a *Party* to engage in electronic *Business* with other *Parties*. These capabilities include both technology capabilities, such as supported communication and messaging protocols, and *Business* capabilities in terms of what *Business Collaborations* it supports.

This section defines and discusses the details in the *CPP* in terms of the individual XML elements. The discussion is illustrated with some XML fragments. See Appendix D for the XML Schema, and Appendix A for sample *CPP* documents.

The ***ProcessSpecification, DeliveryChannel***, ***DocExchange***, and ***Transport*** elements of the *CPP* describe the processing of a unit of *Business* (conversation). These elements form a layered structure somewhat analogous to a layered communication model.

Process-Specification layer - The *Process-Specification* layer defines the heart of the *Business* agreement between the *Parties*: the services (*Business Transaction*s) which *Parties* to the *CPA* can request of each other and transition rules that determine the order of requests. This layer is defined by the separate *Process-Specification* document that is referenced by the *CPP* and *CPA*.

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. Several delivery channels MAY be defined in one *CPP*.

Document-Exchange Layer - 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. The protocol for exchanging *Messages* between two *Parties* is defined by the ebXML Message Service specification[ebMS] or other similar messaging services.

Transport layer - 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.)

Note that the functional layers encompassed by the *CPP* are independent of the contents of the payload of the *Business* documents.

8.1 Globally-Unique Identifier of CPP Instance Document

When a *CPP* is placed in an ebXML or other Registry, the Registry assigns it a globally unique identifier (GUID) that is part of its metadata. That GUID MAY be used to distinguish among *CPPs* belonging to the same *Party*.

NOTE: A Registry cannot insert the GUID into the *CPP*. In general, a Registry does not alter the content of documents submitted to it. Furthermore, a *CPP* MAY be signed and alteration of a signed *CPP* would invalidate the signature.

8.2 CPP Structure

Following is the overall structure of the *CPP*. Unless otherwise noted, *CPP* elements MUST be in the order shown here. Subsequent sections describe each of the elements in greater detail.

```
      <tp:CollaborationProtocolProfile
          xmlns:tp="http://www.oasis-open.org/committees/ebxml-
cppa/schema/cpp-cpa-2_0.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-
cppa/schema/cpp-cpa-2_0.xsd http://www.oasis-open.org/committees/ebxml-
cppa/schema/cpp-cpa-2_0.xsd"
          xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
          xmlns:xlink="http://www.w3.org/1999/xlink"
          tp:cppid="uri:companyA-cpp"
          tp:version="2_0b">
          <tp:PartyInfo>  <!-- one or more -->
          ...
          </tp:PartyInfo>
          <tp:SimplePart id="...">  <!-- one or more -->
          ...
          </tp:SimplePart>
          <tp:Packaging id="...">   <!-- one or more -->
          ...
          </tp:Packaging>
          <tp:Signature>  <!-- zero or one -->
          ...
          </tp:Signature>
          <tp:Comment>text</tp:Comment>  <!-- zero or more -->
      </tp:CollaborationProtocolProfile>
```

8.3 CollaborationProtocolProfile element

The ***CollaborationProtocolProfile*** element is the root element of the *CPP* XML document.

The REQUIRED XML [XML] Namespace[XMLNS] declarations for the basic document are as follows:

- The *CPP/CPA* namespace: xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd",
- The XML Digital Signature namespace: xmlns:ds="http://www.w3.org/2000/09/xmldsig#",
- and the XLink namespace: xmlns:xlink="http://www.w3.org/1999/xlink".

In addition, the ***CollaborationProtocolProfile*** element contains a REQUIRED ***cppid*** attribute that supplies a unique identifier for the document, plus a REQUIRED ***version*** attribute that indicates the version of the schema. Its purpose is to identify the version of the schema that the *CPP* conforms to. The value of the ***version*** attribute SHOULD be a string such as "2_0a", "2_0b", etc.

NOTE: The method of assigning unique cppid values is left to the implementation.

The ***CollaborationProtocolProfile*** element SHALL consist of the following child elements:

- One or more REQUIRED ***PartyInfo*** elements that identify the organization (or parts of the organization) whose capabilities are described by the *CPP*,
- One or more REQUIRED ***SimplePart*** elements that describe the constituents used to make up composite *Messages*,
- One or more REQUIRED ***Packaging*** elements that describe how the *Message Header* and payload constituents are packaged for transmittal,
- Zero or one ***Signature*** element that contains the digital signature that signs the *CPP* document,
- Zero or more ***Comment*** elements.
-

A *CPP* document MAY be digitally signed so as to provide for a means of ensuring that the document has not been altered (integrity) and to provide for a means of authenticating the author of the document. A digitally signed *CPP* SHALL be signed using technology that conforms to the joint W3C/IETF XML Digital Signature specification[XMLDSIG].

8.4 PartyInfo Element

The ***PartyInfo*** element identifies the organization whose capabilities are described in this *CPP* and includes all the details about this *Party*. More than one ***PartyInfo*** element MAY be provided in a *CPP* if the organization chooses to represent itself as subdivisions with different characteristics. Each of the sub-elements of ***PartyInfo*** is discussed later. The overall structure of the ***PartyInfo*** element is as follows:

```
<tp:PartyInfo
    tp:partyName="..."
    tp:defaultMshChannelId="..."
    tp:defaultMshPackageId="...">
    <tp:PartyId tp:type="...">  <!-- one or more -->
    ...
    </tp:PartyId>
    <tp:PartyRef xlink:href="..."/>
    <tp:CollaborationRole>  <!-- one or more -->
    ...
    </tp:CollaborationRole>
    <tp:Certificate>  <!-- one or more -->
    ...
    </tp:Certificate>
    <tp:SecurityDetails>  <!-- one or more -->
    ...
    </tp:SecurityDetails>
    <tp:DeliveryChannel>  <!-- one or more -->
```

```
            ...
            </tp:DeliveryChannel>
            <tp:Transport>  <!-- one or more -->
            ...
            </tp:Transport>
            <tp:DocExchange>  <!-- one or more -->
            ...
            </tp:DocExchange>
            <tp:OverrideMshActionBinding>  <!-- zero or more -->
            ...
            </tp:OverrideMshActionBinding>
        </tp:PartyInfo>
```

The ***PartyInfo*** element contains a REQUIRED ***partyName*** attribute that indicates the common, human readable name of the organization. Unlike ***PartyID***, ***partyName*** might not be unique; however, the value of each ***partyName*** attribute SHALL be meaningful enough to directly identify the organization or the subdivision of an organization described in the ***PartyInfo*** element.

The following example illustrates two possible party names.

```
        <tp:PartyInfo tp:partyName="Example, Inc."...</tp:PartyInfo>

        <tp:PartyInfo tp:partyName="Example, Inc. US Western Division">
        ...
        </tp:PartyInfo>
```

The ***PartyInfo*** element also contains a REQUIRED ***defaultMshChannelId*** attribute and a REQUIRED ***defautMshPackageId*** attribute. The ***defaultMshChannelId*** attribute identifies the default ***DeliveryChannel*** to be used for sending standalone *Message* Service Handler[ebMS] level messages (i.e., Acknowledgment, Error, StatusRequest, StatusResponse, Ping, Pong) that are to be delivered asynchronously. When synchronous reply mode is in use, *Message* Service Handler level messages are by default returned synchronously. The default can be overridden through the use of ***OverrideMshActionBinding*** elements. The ***defaultMshPackageId*** attribute identifies the default Packaging to be used for sending standalone Message Service Handler[ebMS] level messages.

The ***PartyInfo*** element consists of the following child elements:

- One or more REQUIRED ***PartyId*** elements that provide logical identifiers for the organization.
- One or more REQUIRED ***PartyRef*** elements that provide pointers to more information about the *Party*.
- One or more REQUIRED ***CollaborationRole*** elements that identify the roles that this *Party* can play in the context of a *Process Specification*.
- One or more REQUIRED ***Certificate*** elements that identify the certificates used by this *Party* in security functions.
- One or more REQUIRED ***SecurityDetails*** elements that identify trust anchors and specify security policy used by this *Party* in security functions.
- One or more REQUIRED ***DeliveryChannel*** elements that define the characteristics that the *Party* can use to send and/or receive *Messages*. It includes both the transport protocol

(e.g. HTTP) and the messaging protocol (e.g. ebXML *Message* Service).

- One or more REQUIRED ***Transport*** elements that define the characteristics of the transport protocol(s) that the *Party* can support to send and/or receive *Messages*.
- One or more REQUIRED ***DocExchange*** elements that define the *Message*-exchange characteristics, such as the signature and encryption protocols, that the *Party* can support.
- Zero or more ***OverrideMshActionBinding*** elements that specify the DeliveryChannel to use for asynchronously delivered *Message Service Handler* level messages.

8.4.1 PartyId element

The REQUIRED ***PartyId*** element provides an identifier that SHALL be used to logically identify the *Party*. Additional ***PartyId*** elements MAY be present under the same ***PartyInfo*** element so as to provide for alternative logical identifiers for the *Party*. If the *Party* has preferences as to which logical identifier is used, the ***PartyId*** elements SHOULD be listed in order of preference starting with the most-preferred identifier.

In a *CPP* that contains multiple ***PartyInfo*** elements, different ***PartyInfo*** elements MAY contain ***PartyId*** elements that define different logical identifiers. This permits a large organization, for example, to have different identifiers for different purposes.

The value of the ***PartyId*** element is any string that provides a unique identifier. The identifier MAY be any identifier that is understood by both *Parties* to a *CPA*. Typically, the identifier would be listed in a well-known directory such as DUNS (Dun and Bradstreet) or in any naming system specified by [ISO6523].

The ***PartyId*** element has a single IMPLIED attribute: ***type*** that has an anyURI [XMLSCHEMA-2] value.

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]. It is RECOMMENDED that the value of the ***type*** attribute be a URN that defines a namespace for the value of the ***PartyId*** element. Typically, the URN would be registered in a well-known directory of organization identifiers.

The following example illustrates two URI references.

```
  <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-
type:duns">123456789</tp:PartyId>

  <tp:PartyId>urn:icann:example.com</tp:PartyId>
```

The first example is the *Party's* DUNS number. The value is the DUNS number of the organization.

The second example shows an arbitrary URN. This might be a URN that the *Party* has registered with IANA, the Internet Assigned Numbers Authority (http://www.iana.org) to identify itself directly.

The following document discusses naming agencies and how they are identified via URI values of the ***type*** attribute:

http://www.oasis-open.org/committees/ebxml-cppa/documents/PartyID_Types.shtml

8.4.2 PartyRef element

The ***PartyRef*** element provides a link, in the form of a URI, to additional information about the *Party*. Typically, this would be the URL from which the information can be obtained. The information might be at the *Party's* web site or in a publicly accessible repository such as an ebXML Registry, a UDDI repository (www.uddi.org), or a Lightweight Directory Access Protocol[RFC2251] (LDAP) directory. Information available at that URI MAY include contact information like names, addresses, and phone numbers, or context information like geographical locales and industry segments, or perhaps more information about the *Business Collaborations* that the *Party* supports. This information MAY be in the form of an ebXML Core Component[ccOVER]. It is not within the scope of this specification to define the content or format of the information at that URI.

The ***PartyRef*** element is an [XLINK] simple link. It has the following attributes:

- a FIXED ***xlink:type*** attribute,
- a REQUIRED ***xlink:href*** attribute,
- an IMPLIED ***type*** attribute,
- an IMPLIED ***schemaLocation*** attribute.

The contents of the document referenced by the ***partyRef*** element are subject to change at any time. Therefore, it SHOULD NOT be cached for a long period of time. Rather, the value of the xlink:href SHOULD be dereferenced only when the contents of this document are needed.

8.4.2.1 xlink:type attribute

The FIXED ***xlink:type*** attribute SHALL have a value of "simple". This identifies the element as being an [XLINK] simple link.

8.4.2.2 xlink:href attribute

The REQUIRED ***xlink:href*** attribute SHALL have a value that is a URI that conforms to [RFC2396] and identifies the location of the external information about the *Party*.

8.4.2.3 type attribute

The value of the IMPLIED ***type*** attribute identifies the document type of the external information about the *Party*. It MUST be a URI that defines the namespace associated with the information about the *Party*. If the ***type*** attribute is omitted, the external information about the *Party* MUST be an HTML web page.

8.4.2.4 schemaLocation attribute

The value of the IMPLIED ***schemaLocation*** attribute provides a URI for the schema that describes the structure of the external information.

An example of the ***PartyRef*** element is:

```
<tp:PartyRef xlink:type="simple"
    xlink:href="http://example2.com/ourInfo.xml"
    tp:type="urn:oasis:names:tc:ebxml-cppa:contact-info"
    tp:schemaLocation="http://example2.com/ourInfo.xsd"/>
```

8.4.3 CollaborationRole element

The ***CollaborationRole*** element associates a *Party* with a specific role in the *Business Collaboration*. Generally, the *Process-Specification* is defined in terms of roles such as "buyer" and "seller". The association between a specific *Party* and the role(s) it is capable of fulfilling within the context of a *Process-Specification* is defined in both the *CPP* and *CPA* documents. In a *CPP*, the ***CollaborationRole*** element identifies which role the *Party* is capable of playing in each *Process Specification* documents referenced by the *CPP*. An example of the ***CollaborationRole*** element, based on RosettaNet™ PIP 3A4 is:

```
<tp:CollaborationRole >
    <tp:ProcessSpecification
        tp:version="2.0a"
        tp:name="PIP3A4RequestPurchaseOrder"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml"/>
    <tp:Role
        tp:name="Buyer"
        xlink:type="simple"

xlink:href="http://www.rosettanet.org/processes/3A4.xml#BuyerId"/>
    <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
    <tp:ServiceBinding>
        <tp:Service
tp:type="anyURI">urn::icann:rosettanet.org:bpid:3A4$2.0</tp:Service>
        <tp:CanSend>
            <tp:ThisPartyActionBinding
                tp:id="companyA_ABID1"
                tp:action="Purchase Order Request Action"
                tp:packageId="CompanyA_RequestPackage">
                <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"/>
                    <tp:ChannelId>asyncChannelA1</tp:ChannelId>
                </tp:ThisPartyActionBinding>
            </tp:CanSend>
            <tp:CanSend>
                <tp:ThisPartyActionBinding
                    tp:id="companyA_ABID2"
                    tp:action="ReceiptAcknowledgment"
                    tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
                    <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:ChannelId>asyncChannelA1</tp:ChannelId>
                </tp:ThisPartyActionBinding>
            </tp:CanSend>
            <tp:CanReceive>
                <tp:ThisPartyActionBinding
                    tp:id="companyA_ABID3"
                    tp:action="Purchase Order Confirmation Action"
                    tp:packageId="CompanyA_ResponsePackage">
                    <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
Confirmation Action"/>
                    <tp:ChannelId>asyncChannelA1</tp:ChannelId>
                </tp:ThisPartyActionBinding>
            </tp:CanReceive>
            <tp:CanReceive>
                <tp:ThisPartyActionBinding
                    tp:id="companyA_ABID4"
                    tp:action="ReceiptAcknowledgment"
                    tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
                    <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:ChannelId>asyncChannelA1</tp:ChannelId>
                </tp:ThisPartyActionBinding>
            </tp:CanReceive>
            <tp:CanReceive>
                <tp:ThisPartyActionBinding
```

```
                        tp:id="companyA_ABID5"
                        tp:action="Exception"
                        tp:packageId="CompanyA_ExceptionPackage">
                        <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:ChannelId>asyncChannelA1</tp:ChannelId>
                    </tp:ThisPartyActionBinding>
                </tp:CanReceive>
            </tp:ServiceBinding>
        </tp:CollaborationRole>
```

To indicate that the *Party* can play roles in more than one *Business Collaboration* or more than one role in a given *Business Collaboration*, the ***PartyInfo*** element SHALL contain more than one ***CollaborationRole*** element. Each ***CollaborationRole*** element SHALL contain the appropriate combination of ***ProcessSpecification*** element and ***Role*** element.

The ***CollaborationRole*** element SHALL consist of the following child elements: a REQUIRED ***ProcessSpecification*** element, a REQUIRED ***Role*** element, zero or one ***ApplicationCertificateRef*** elements, zero or one ***ApplicationSecurityDetailsRef*** element, and one ***ServiceBinding*** element. The ***ProcessSpecification*** element identifies the *Process-Specification* document that defines such role. The ***Role*** element identifies which role the *Party* is capable of supporting. The ***ApplicationCertificateRef*** element identifies the certificate to be used for application level signature and encryption. The ***ApplicationSecurityDetailsRef*** element identifies the trust anchors and security policy that will be applied to any application-level certificate offered by the other *Party*. The ***ServiceBinding*** element SHALL consist of zero or more ***CanSend*** elements and zero or more ***CanReceive*** elements. The ***CanSend*** and ***CanReceive*** elements identify the ***DeliveryChannel*** elements that are to be used for sending and receiving business action messages by the ***Role*** in question. They MAY also be used for specifying ***DeliveryChannels*** for business signal messages.

Each *Party* SHALL have a default delivery channel for the delivery of standalone *Message* Service Handler level signals like (Reliable Messaging) Acknowledgments, Errors, StatusRequest, StatusResponse, etc.

8.4.4 ProcessSpecification element

The ***ProcessSpecification*** element provides the link to the *Process-Specification* document that defines the interactions between the two *Parties*. It is RECOMMENDED that this *Business-Collaboration* description be prepared in accordance with the ebXML Business Process Specification Schema[ebBPSS]. The *Process-Specification* document MAY be kept in an ebXML Registry.

NOTE: A *Party* can describe the *Business Collaboration* using any desired alternative to

the ebXML Business Process Specification Schema. When an alternative *Business-Collaboration* description is used, the *Parties* to a *CPA* MUST agree on how to interpret the *Business-Collaboration* description and how to interpret the elements in the *CPA* that reference information in the *Business-Collaboration* description. The affected elements in the *CPA* are the ***Role*** element, the ***CanSend*** and ***CanReceive*** elements, the ***ActionContext*** element, and some attributes of the ***BusinessTransactionCharacteristics*** element.

The syntax of the ***ProcessSpecification*** element is:

```
<tp:ProcessSpecification
    tp:version="2.0a"
    tp:name="PIP3A4RequestPurchaseOrder"
    xlink:type="simple"
    xlink:href="http://www.rosettanet.org/processes/3A4.xml"
    uuid="urn:icann:rosettanet.org:bpid:3A4$2.0">
    <ds:Reference ds:URI="http://www.rosettanet.org/processes/3A4.xml">
        <ds:Transforms>
            <ds:Transform
ds:Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        </ds:Transforms>
        <ds:DigestMethod
            ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
    </ds:Reference>
</tp:ProcessSpecification>
```

The ***ProcessSpecification*** element has zero or more child ***ds:Reference*** elements**,** and the following attributes:

- a REQUIRED ***name*** attribute,
- a REQUIRED ***version*** attribute,
- a FIXED ***xlink:type*** attribute,
- a REQUIRED ***xlink:href*** attribute,
- an IMPLIED ***uuid*** attribute.

The ***ProcessSpecification*** element contains zero or more ***ds:Reference*** elements formulated according to the XML Digital Signature specification[XMLDSIG]. The first ***ds:Reference*** element, if present, relates to the ***xlink:type*** and ***xlink:href*** attributes as follows. Each ***ProcessSpecification*** element SHALL contain one ***xlink:href*** attribute and one ***xlink:type*** attribute with a value of "simple". In case the *CPP* (*CPA*) document is signed, the first ***ds:Reference*** element that is present MUST include a ***ds:URI*** attribute whose value is identical to that of the ***xlink:href*** attribute in the enclosing ***ProcessSpecification*** element. The ***ds:Reference*** element specifies a digest method and digest value to enable verification that the referenced *Process-Specification* document has not changed. Additional ***ds:Reference*** elements are needed if the referenced ***ProcessSpecification*** in turn includes (i.e., references) other ***ProcessSpecification***s. Essentially, ***ds:Reference*** elements MUST be provided to correspond to the transitive closure of all ***ProcessSpecification***s that are referenced directly or indirectly to ensure that none of them has been changed.

8.4.4.1 name attribute

The ***ProcessSpecification*** element MUST include a REQUIRED ***name*** attribute: a string that identifies the *Business Process-Specification* being performed. If the *Process-Specification* document is defined by the ebXML Business Process specification [ebBPSS], then this attribute MUST be set to the ***name*** for the corresponding ***ProcessSpecification*** element within the *Business Process Specification* instance.

8.4.4.2 version attribute

The ***ProcessSpecification*** element includes a REQUIRED ***version*** attribute to indicate the version of the *Process-Specification* document identified by the ***xlink:href*** attribute (and also identified by the ***ds:Reference*** element, if any).

8.4.4.3 xlink:type attribute

The ***xlink:type*** attribute has a FIXED value of "simple". This identifies the element as being an [XLINK] simple link.

8.4.4.4 xlink:href attribute

The REQUIRED ***xlink:href*** attribute SHALL have a value that identifies the *Process-Specification* document and is a URI that conforms to [RFC2396].

8.4.4.5 uuid attribute

The IMPLIED ***uuid*** attribute uniquely identifies the ***ProcessSpecification***. If the *Process-Specification* document is defined by the ebXML Business Process specification [ebBPSS], then this attribute MUST be set to the ***uuid*** for the corresponding ***ProcessSpecification*** element within the business process specification instance.

8.4.4.6 ds:Reference element

The ***ds:Reference*** element identifies the same *Process-Specification* document as the enclosing ***ProcessSpecification*** element's ***xlink:href*** attribute and additionally provides for verification that the *Process-Specification* document has not changed since the *CPP* was created, through the use of a digest method and digest value as described below.

> NOTE: *Parties* MAY test the validity of the *CPP* or *CPA* at any time. The following validity tests MAY be of particular interest:
>
> - test of the validity of a *CPP* and the referenced *Process-Specification* documents at the time composition of a *CPA* begins in case they have changed since they were created,
> - test of the validity of a *CPA* and the referenced *Process-Specification* documents at the time a *CPA* is installed into a *Party's* system,
> - test of the validity of a *CPA* at intervals after the *CPA* has been installed into a *Party's* system. The *CPA* and the referenced *Process-Specification* documents MAY be processed by an installation tool into a form suited to the particular middleware. Therefore, alterations to the *CPA* and the referenced *Process-Specification* documents do not necessarily affect ongoing run-time operations. Such alterations might not be

detected until it becomes necessary to reinstall the *CPA* and the referenced *Process-Specification* documents.

The syntax and semantics of the ***ds:Reference*** element and its child elements are defined in the XML Digital Signature specification[XMLDSIG]. In addition, to identify the *Process-Specification* document, the first ***ds:Reference*** element MUST include a ***ds:URI*** attribute whose value is identical to that of the ***xlink:href*** attribute in the enclosing ***ProcessSpecification*** element.

According to [XMLDSIG], a ***ds:Reference*** element can have a ***ds:Transforms*** child element, which in turn has an ordered list of one or more ***ds:Transform*** child elements to specify a sequence of transforms. However, this specification currently REQUIRES the Canonical XML[XMLC14N] transform and forbids other transforms. Therefore, the following additional requirements apply to a ***ds:Reference*** element within a ***ProcessSpecification*** element:

- The ***ds:Reference*** element MUST have a ***ds:Transforms*** child element.
- That ***ds:Transforms*** element MUST have exactly one ***ds:Transform*** child element.
- That ***ds:Transform*** element MUST specify the Canonical XML[XMLC14N] transform via the following REQUIRED value for its REQUIRED ***ds:Algorithm*** attribute: `http://www.w3.org/TR/2001/Rec-xml-c14n-20010315.`

  Note that implementation of Canonical XML is REQUIRED by the XML Digital Signature specification[XMLDSIG].

To enable verification that the identified and transformed *Process-Specification* document has not changed, the ***ds:DigestMethod*** element specifies the digest algorithm applied to the *Process-Specification* document, and the ***ds:DigestValue*** element specifies the expected value. The *Process-Specification* document is presumed to be unchanged if and only if the result of applying the digest algorithm to the *Process-Specification* document results in the expected value.

A ***ds:Reference*** element in a ***ProcessSpecification*** element has implications for *CPP* validity:

- A *CPP* MUST be considered invalid if any ***ds:Reference*** element within a ***ProcessSpecification*** element fails reference validation as defined by the XML Digital Signature specification[XMLDSIG].

- A *CPP* MUST be considered invalid if any ***ds:Reference*** element within it cannot be dereferenced.

Other validity implications of such ***ds:Reference*** elements are specified in the description of the ***Signature*** element in Section 9.9.

NOTE: The XML Digital Signature specification[XMLDSIG] states "The signature application MAY rely upon the identification (URI) and Transforms provided by the signer in the Reference element, or it MAY obtain the content through other means such

as a local cache" (emphasis on MAY added). However, it is RECOMMENDED that ebXML *CPP*/*CPA* implementations not make use of such cached results when signing or validating.

NOTE: It is recognized that the XML Digital Signature specification[XMLDSIG] provides for signing an XML document together with externally referenced documents. In cases where a *CPP* or *CPA* document is in fact suitably signed, that facility could also be used to ensure that the referenced *Process-Specification* documents are unchanged. However, this specification does not currently mandate that a *CPP* or *CPA* be signed.

NOTE: If the *Parties* to a *CPA* wish to customize a previously existing *Process-Specification* document, they MAY copy the existing document, modify it, and cause their *CPA* to reference the modified copy. It is recognized that for reasons of clarity, brevity, or historical record, the *Parties* might prefer to reference a previously existing *Process-Specification* document in its original form and accompany that reference with a specification of the agreed modifications. Therefore, *CPP* usage of the ***ds:Reference*** element's ***ds:Transforms*** sub-element within a ***ProcessSpecification*** element might be expanded in the future to allow other transforms as specified in the XML Digital Signature specification[XMLDSIG]. For example, modifications to the original document could then be expressed as XSLT transforms. After applying any transforms, it would be necessary to validate the transformed document against the ebXML Business Process Specification Schema[ebBPSS].

8.4.5 Role element

The REQUIRED ***Role*** element identifies which role in the *Process Specification* the *Party* is capable of supporting via the ***ServiceBinding*** element(s) siblings within this ***CollaborationRole*** element.

The ***Role*** element has the following attributes:

- a REQUIRED ***name*** attribute,
- a FIXED ***xlink:type*** attribute,
- a REQUIRED ***xlink:href*** attribute.

8.4.5.1 name attribute

The REQUIRED ***name*** attribute is a string that gives a name to the ***Role***. Its value is taken from a name attribute of one of a ***BinaryCollaboration***'s ***Role*** elements described in the *Process Specification*[ebBPSS].

See NOTE in Section 8.4.4 regarding alternative *Business-Collaboration* descriptions.

8.4.5.2 xlink:type attribute

The ***xlink:type*** attribute has a FIXED value of "simple". This identifies the element as being an [XLINK] simple link.

8.4.5.3 xlink:href attribute

The REQUIRED ***xlink:href*** attribute SHALL have a value that is a URI that conforms to [RFC2396]. It identifies the location of the element or attribute within the *Process-Specification* document that defines the role in the context of the *Business Collaboration.* An example is:

```
xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"
```

Where "Buyer" is the value of the ID attribute of the element in the *Process-Specification* document that defines the role name.

8.4.6 ApplicationCertificateRef element

The ***ApplicationCertificateRef*** element, if present, identifies a certificate for use by the business process/application layer. This certificate is not used by the ebXML messaging system, but it is included in the *CPP* so that it can be considered in the *CPA* negotiation process. The ***ApplicationCertificateRef*** element can occur zero or more times.

NOTE: It is up to the application software on both sides of a collaboration to determine the intended/allowed usage of an application certificate by inspecting the key usage extension within the certificate itself.

NOTE: This element is included in the *CPP/CPA* to support interoperability with legacy systems that already perform cryptographic functions such as digital signature or encryption. Implementers should understand that use of ***ApplicationCertificateRef*** is necessary only in cases where interoperability with such legacy systems is required.

The ***ApplicationCertificateRef*** element has

- A REQUIRED ***certId*** attribute.

8.4.6.1 certId attribute

The REQUIRED ***certId*** attribute is an [XML] IDREF that associates the ***CollaborationRole*** element with a certificate**.** It MUST have a value equal the value of the ***certId*** attribute of one of the ***Certificate*** elements under ***PartyInfo***.

8.4.7 ApplicationSecurityDetailsRef element

The ***ApplicationSecurityDetailsRef*** element, if present, identifies the trust anchors and security policy that this *Party* will apply to any application-level certificate offered by the other *Party*. These trust anchors and policy are not used by the ebXML messaging system, but are included in the *CPP* so that they can be considered in the *CPA* negotiation process.

The ***ApplicationSecurityDetailsRef*** element has

- A REQUIRED ***securityId*** attribute.
-

8.4.7.1 SecurityId attribute

The REQUIRED ***securityId*** attribute is an [XML] IDREF that associates the ***CollaborationRole*** with a ***SecurityDetails*** element that specifies a set of trust anchors and a security policy. It MUST have a value equal to the value of the ***securityId*** attribute of one of the ***SecurityDetails*** elements under ***PartyInfo***.

8.4.8 ServiceBinding element

The ***ServiceBinding*** element identifies a ***DeliveryChannel*** element for all of the business *Message* traffic that is to be sent or received by the *Party* within the context of the identified *Process-Specification* document. It MUST contain at least one ***CanReceive*** or ***CanSend*** child element.

The ***ServiceBinding*** element has one child ***Service*** element, zero or more ***CanSend*** child elements, and zero or more ***CanReceive*** child elements.

8.4.9 Service element

The value of the ***Service*** element is a string that SHALL be used as the value of the ***Service*** element in the ebXML *Message Header*[ebMS] or a similar element in the *Message Header* of an alternative *message* service. The ***Service*** element has an IMPLIED ***type*** attribute.

If the *Process-Specification* document is defined by the ebXML Business Process Specification Schema[ebBPSS], then the value of the ***Service*** element MUST be the ***uuid*** (URI) attribute specified for the ***ProcessSpecification*** element in the Business Process Specification Schema instance document.

> NOTE: The purpose of the ***Service*** element is to provide routing information for the ebXML *Message Header*. The ***CollaborationRole*** element and its child elements identify the information in the ***ProcessSpecification*** document that is relevant to the *CPP* or *CPA*. The ***Service*** element MAY be used along with the ***CanSend*** and ***CanReceive*** elements (and their descendants) to provide routing of received messages to the correct application entry point.

8.4.9.1 type attribute

If the ***type*** attribute is present, it indicates that the *Parties* sending and receiving the *Message* know, by some other means, how to interpret the value of the ***Service*** element. The two *Parties* MAY use the value of the ***type*** attribute to assist the interpretation.

If the ***type*** attribute is not present, the value of the ***Service*** element MUST be a URI[RFC2396]. If using the ebXML Business Process Specification[ebBPSS] for defining the *Process-Specification* document, the type attribute MUST be a URI[RFC2396].

8.4.10 CanSend element

The ***CanSend*** element identifies an *action* message that a *Party* is capable of sending. It has

three sub-elements: ***ThisPartyActionBinding, OtherPartyActionBinding,*** and ***CanReceive***. The ***ThisPartyActionBinding*** element is REQUIRED for both *CPPs* and *CPAs*. It identifies the ***DeliveryChannel*** and the ***Packaging*** the *Party* described by the encompassing ***PartyInfo*** element will use for sending the *action* invocation message in question. The ***OtherPartyActionBinding*** element is only used in the case of *CPAs*. Within a *CPA* and under the same ***CanSend*** element, the ***DeliveryChannels*** and ***Packaging*** used/expected by the two *Parties* MUST be compatible. The ***CanReceive*** element can occur zero or more times. When present, it indicates that one or more synchronous response actions are expected.
This is illustrated in the *CPP* and *CPA* examples in the appendices.

NOTE: While the schema permits arbitrary nesting levels under the ***CanSend*** element, use cases for nesting beyond two levels have not yet been presented. Two levels could be needed for a Request with a synchronously returned Response that additionally specified a synchronously returned Acknowledgment for that Response.

8.4.11 CanReceive element

The ***CanReceive*** element identifies an *action* invocation message that a *Party* is capable of receiving. It has three sub-elements: ***ThisPartyActionBinding***, ***OtherPartyActionBinding,*** and ***CanSend***. The ***ThisPartyActionBinding*** element is REQUIRED for both *CPPs* and *CPAs*. It identifies the ***DeliveryChannel*** the *Party* described by the encompassing ***PartyInfo*** element will use for receiving the *action* message in question and the ***Packaging*** it is expecting. The ***OtherPartyActionBinding*** element is only used in the case of CPAs. Within a *CPA* and under the same ***CanReceive*** element, the ***DeliveryChannels*** and ***Packaging*** used/expected by the two *Parties* MUST be compatible. The ***CanSend*** element can occur zero or more times. When present, it indicates that one or more synchronous response actions are expected. This is illustrated in the *CPP* and *CPA* examples in the appendices.

NOTE: While the schema permits arbitrary nesting levels under the ***CanReceive*** element, use cases for nesting beyond two levels have not yet been presented. Two levels could be needed for a Request with a synchronously returned Response that additionally specified a synchronously returned Acknowledgment for that Response.

8.4.12 ThisPartyActionBinding element

The ***ThisPartyActionBinding*** specifies one or more ***DeliveryChannel*** elements for *Messages* for a selected *action* and the ***Packaging*** for those *Messages* that are to be sent or received by the *Party* in the context of the *Process Specification* that is associated with the parent ***ServiceBinding*** element.

The ***ThisPartyActionBinding*** element has a REQUIRED child ***BusinessTransactionCharacteristics*** element, zero or one child ***ActionContext*** element and one or more ***ChannelID*** child elements.

The ***ThisPartyActionBinding*** element has the following attributes:

- a REQUIRED ***action*** attribute,
- a REQUIRED ***packageId*** attribute,
- an IMPLIED ***xlink:href*** attribute,
- a FIXED ***xlink:type*** attribute.

Under a given ***ServiceBinding*** element, there MAY be multiple ***CanSend*** or ***CanReceive*** child elements with the same ***action*** to allow different software entry points and Transport options. In such a scenario, the ***DeliveryChannels*** referred by the ***ChannelID*** child elements of ***ThisPartyActionBinding*** SHALL point to distinct ***EndPoint***s for the receiving MSH to uniquely identify the ***DeliveryChannel*** being used for this particular message exchange.

> NOTE: An implementation MAY provide the capability of dynamically assigning delivery channels on a per *Message* basis during performance of the ***BinaryCollaboration***. The delivery channel selected would be chosen, based on present conditions, from those identified by ***CanSend*** elements that refer to the ***BinaryCollaboration*** that is sending the *Message*. On the receiving side, the MSH can use the distinct ***EndPoints*** to identify the ***DeliveryChannel*** used for this message exchange.

Within a ***CanSend*** element or a ***CanReceive*** element, when both the ***ThisPartyActionBinding*** and ***OtherPartyActionBinding*** elements are present (i.e., in a *CPA*), they MUST have identical action values or equivalent ***ActionContext*** elements. In addition, the ***DeliveryChannel*** and ***Packaging*** that that they reference MUST be compatible.

8.4.12.1 action attribute

The value of the REQUIRED ***action*** attribute is a string that identifies the business document exchange to be associated with the ***DeliveryChannel*** identified by the ***ChannelId*** sub-elements. The value of the ***action*** attribute SHALL be used as the value of the *Action* element in the ebXML *Message Header*[ebMS] or a similar element in the *Message Header* of an alternative *message* service. The purpose of the ***action*** attribute is to provide a mapping between the hierarchical naming associated with a *Business Process/Application* and the *Action* element in the ebXML Message Header[ebMS]. This mapping MAY be implemented by using the ***ActionContext*** element. See NOTE in Section 8.4.4 regarding alternative *Business Collaboration* descriptions.

Business signals, when sent individually (i.e., not bundled with response documents in synchronous reply mode), SHALL use the values *ReceiptAcknowledgment*, *AcceptanceAcknowledgment*, or *Exception* as the value of their ***action*** attribute. In addition, they SHOULD specify a Service that is the same as the Service used for the original message.

> NOTE: In general, the action name chosen by the two *Parties* to represent a particular requesting business activity or responding business activity in the context of a ***Binary Collaboration*** that makes use of nested ***BinaryCollaboration***s MAY not be identical.

Therefore, when composing two *CPPs* to form a *CPA*, it is necessary to make use of information from the associated ***ActionContext*** (see Section 8.4.16) in order to determine if two different action names from the two *CPPs* actually represent the same ***ActionContext***. When business transactions are not reused in different contexts, it is recommended that the names of the requesting business activity and responding business activity be used as action names.

8.4.12.2 packageId attribute

The REQUIRED ***packageId*** attribute is an [XML] IDREF that identifies the ***Packaging*** element to be associated with the *Message* identified by the ***action*** attribute.

8.4.12.3 xlink:href attribute

The IMPLIED ***xlink:href*** attribute, if present, SHALL provide an absolute [XPOINTER] URI expression that specifically identifies the ***RequestingBusinessActivity*** or ***RespondingBusinessActivity*** element within the associated *Process-Specification* document[ebBPSS] that is identified by the ***ProcessSpecification*** element.

8.4.12.4 xlink:type attribute

The IMPLIED ***xlink:type*** attribute has a FIXED value of "simple". This identifies the element as being an [XLINK] simple link.

8.4.13 OtherPartyActionBinding

The ***OtherPartyActionBinding*** element is only used in the case of *CPAs*. It is of type IDREF and identifies a matching ***ThisPartyActionBinding*** element that is found under the collaboration partner's ***PartyInfo***. It indirectly identifies the ***DeliveryChannel*** the other *Party* will use for sending or receiving the *action* message in question and the expected ***Packaging***. Within a *CPA* and under the same ***CanSend*** or ***CanReceive*** element, the ***DeliveryChannels*** and ***Packaging*** used/expected by the two *Parties*, as indicated by the ***ThisPartyActionBinding*** and ***OtherPartyActionBinding*** elements, MUST be compatible.

8.4.14 BusinessTransactionCharacteristics element

The ***BusinessTransactionCharacteristics*** element describes the security characteristics and other attributes of the delivery channel, as derived from the ***ProcessSpecification(s)*** whose messages are transported using the delivery channel. The attributes of the ***BusinessTransactionCharacteristics*** element, MAY be used to override the values of the corresponding attributes in the *Process-Specification* document.

See NOTE in Section 8.4.4 regarding alternative *Business-Collaboration* descriptions.

CPP and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel definitions for the sender and receiver (transport and document-exchange) for internal consistency as well as compatibility between the two partners. Typically, when an attribute has a particular value, sub-elements under the corresponding Transport and DocExchange elements

would exist to further describe the implied implementation parameters.

The ***BusinessTransactionCharacteristics*** element has the following attributes:

- an IMPLIED isNonRepudiationRequired attribute,
- an IMPLIED isNonRepudiationReceiptRequired attribute,
- an IMPLIED ***isConfidential*** attribute,
- an IMPLIED ***isAuthenticated*** attribute,
- an IMPLIED isAuthorizationRequired attribute,
- an IMPLIED ***isTamperProof*** attribute,
- an IMPLIED isIntelligibleCheckRequired attribute,
- an IMPLIED timeToAcknowledgeReceipt attribute,
- an IMPLIED timeToAcknowledgeAcceptance attribute,
- an IMPLIED ***timeToPerform*** attribute,
- an IMPLIED ***retryCount*** attribute.
-

These attributes allow parameters specified at the *Process-Specification* level to be overridden. If one of these attributes is not specified, the corresponding default value should be obtained from the *Process-Specification* document.

8.4.14.1 isNonRepudiationRequired attribute

The ***isNonRepudiationRequired*** attribute is a Boolean with possible values of "true" and "false". If the value is "true" then the delivery channel MUST specify that the *Message* is to be digitally signed using the certificate of the *Party* sending the *Message,* and archived by both *Parties*. The ***SenderNonRepudiation*** element under ***DocExchange***/***ebXMLSenderBinding*** (see Section 8.4.43) and the ***ReceiverNonRepudiation*** element under ***DocExchange***/***ebXMLReceiverBinding*** (see Section 0) further describe various parameters related to the implementation of non-repudiation of origin, such as the hashing algorithm, the signature algorithm, the signing certificate, the trust anchor, etc.

8.4.14.2 isNonRepudiationReceiptRequired attribute

The ***isNonRepudiationReceiptRequired*** attribute is a Boolean with possible values of "true" and "false". If the value is "true" then the delivery channel MUST specify that the *Message* is to be acknowledged by a digitally signed *Receipt Acknowledgment* signal *Message*, signed using the certificate of the *Party* that received the *Message*, that includes the digest(s) of the payload(s) of the *Message* being acknowledged. The ***SenderNonRepudiation*** element under ***DocExchange***/***ebXMLSenderBinding*** (see Section 8.4.43) and the ***ReceiverNonRepudiation*** element under ***DocExchange***/***ebXMLReceiverBinding*** (see Section 0) further describe various parameters related to the implementation of non-repudiation of receipt.

8.4.14.3 isConfidential attribute

The ***isConfidential*** attribute has the possible values of "none", "transient", "persistent", and "transient-and-persistent". These values MUST be interpreted as defined by the ebXML Business Process Specification Schema[ebBPSS]. In general, transient confidentiality can be implemented

using a secure transport protocol like SSL; persistent confidentiality can be implemented using a digital envelope mechanism like S/MIME. Secure transport information is further provided in the ***TransportSender*** (see Section 8.4.25) and ***TransportReceiver*** (see Section 8.4.32) elements under the ***Transport*** element. Persistent encryption information is further provided in the ***SenderDigitalEnvelope*** element under ***DocExchange***/***ebXMLSenderBinding*** (see Section 8.4.48) and the ***ReceiverDigitalEnvelope*** element under ***DocExchange***/***ebXMLReceiverBinding*** (see Section 8.4.56).

8.4.14.4 isAuthenticated attribute

The ***isAuthenticated*** attribute has the possible values of "none", "transient", "persistent", and "persistent-and-transient". If this attribute is set to any value other than "none", then the receiver MUST be able to verify the identity of the sender. In general, transient authentication can be implemented using a secure transport protocol like SSL (with or without the use of basic or digest authentication); persistent authentication can be implemented using a digital signature mechanism. Secure transport information is further provided in the ***TransportSender*** (see Section 8.4.25) and ***TransportReceiver*** (see Section 8.4.33) elements under the ***Transport*** element. Persistent authentication information is further provided in the ***SenderNonRepudiation*** element under ***DocExchange***/***ebXMLSenderBinding*** (see Section 8.4.43) and the ***ReceiverNonRepudiation*** element (under ***DocExchange***/***ebXMLReceiverBinding*** (see Section 0).

8.4.14.5 isAuthorizationRequired attribute

The ***isAuthorizationRequired*** attribute is a Boolean with possible of values of "true" and "false". If the value is "true" then it indicates that the delivery channel MUST specify that the sender of the *Message* is to be authorized before delivery to the application.

8.4.14.6 isTamperProof attribute

The ***isTamperProof*** attribute has the possible values of "none", "transient", "persistent", and "persistent-and-transient". If this attribute is set to a value other than "none", then it must be possible for the receiver to detect if the received message has been corrupted or tampered with. In general, transient tamper detection can be implemented using a secure transport like SSL; persistent tamper detection can be implemented using a digital signature mechanism. Secure transport information is further provided in the ***TransportSender*** (see Section 8.4.25) and ***TransportReceiver*** (see Section 8.4.48) elements under the ***Transport*** element. Digital signature information is further provided in the ***SenderNonRepudiation*** element under ***DocExchange***/***ebXMLSenderBinding*** (see Section 8.4.43) and the ***ReceiverNonRepudiation*** element under ***DocExchange***/***ebXMLReceiverBinding*** (see Section 0).

8.4.14.7 isIntelligibleCheckRequired attribute

The ***isIntelligibleCheckRequired*** attribute is a Boolean with possible values of "true" and "false". If the value is "true", then the receiver MUST verify that a business document is not garbled (i.e., passes schema validation) before returning a *Receipt Acknowledgment* signal.

8.4.14.8 timeToAcknowledgeReceipt attribute

The ***timeToAcknowledgeReceipt*** attribute is of type duration [XMLSCHEMA-2]. It specifies the time period within which the receiving *Party* has to acknowledge receipt of a business document.

If this attribute is specified, then the *Receipt Acknowledgment* signal MUST be used.

8.4.14.9 timeToAcknowledgeAcceptance attribute

The ***timeToAcknowledgeAcceptance*** attribute is of type duration [XMLSCHEMA-2]. It specifies the time period within which the receiving *Party* has to non-substantively acknowledge acceptance of a business document (i.e., after it has passed business rules validation).
If this attribute is specified, then the *Acceptance Acknowledgment* signal MUST be used.

8.4.14.10 timeToPerform attribute

The ***timeToPerform*** attribute is of type duration [XMLSCHEMA-2]. It specifies the time period, starting from the initiation of the ***RequestingBusinessActivity***, within which the initiator of the transaction MUST have received the response, i.e., the business document associated with the ***RespondingBusinessActivity***.

> NOTE: The ***timeToPerform*** attribute associated with a ***BinaryCollaboration*** in BPSS is currently not modeled in this specification. Therefore, it cannot be overridden. In other words, the value specified at the BPSS level MUST be used.

When synchronous reply mode is in use (see Section 8.4.23.1), the ***TimeToPerform*** value SHOULD be used as the connection timeout.

8.4.14.11 retryCount attribute

The ***retryCount*** attribute is of type integer. It specifies the maximum number of times the *Business Transaction* MAY be retried should certain error conditions (e.g., time out waiting for the Receipt Acknowledgment signal) arise during its execution. Such retries MUST not be used when ebXML Reliable Messaging is employed to transport messages in the *Business Transaction*. In the latter case, retries are governed by the ***Retry***, ***RetryInterval*** elements under the ***ReliableMessaging*** element.

8.4.15 ChannelId element

The ***ChannelId*** element identifies one or more ***DeliveryChannel*** elements that can be used for sending or receiving the corresponding action messages. Multiple ***ChannelId*** elements can be used to associate ***DeliveryChannel*** elements with different characteristics with the same ***CanSend*** or ***CanReceive*** element. For example, a *Party* that supports both HTTP and SMTP for sending the same action can specify different ***ChannelId*** attribute values for the corresponding channels. If using multiple ***DeliveryChannel*** elements**,** different ***EndPoint*** elements MUST be used, so that the receiving MSH can uniquely determine the ***DeliveryChannel*** element being used for this message exchange.

8.4.16 ActionContext element

The ***ActionContext*** element provides a mapping from the ***action*** attribute in the ***ThisPartyActionBinding*** element to the corresponding *Business Process* implementation-specific naming strategy, if any. If the *Process-Specification* document is defined by the ebXML Business Process Specification Schema[ebBPSS], the ***ActionContext*** element MUST be present.

Any business process/application implementation can use a combination of information in the ***action*** attribute and the ***ActionContext*** elements to make message routing decisions. If using alternative *Business-Collaboration* description schemas, the ***action*** attribute of the parent ***ThisPartyActionBinding*** element and/or the [XMLSCHEMA-1] ***wildcard*** element within the ***ActionContext*** element MAY be used to make routing decisions above the level of the *Message* Service Handler.

The ***ActionContext*** element has the following elements:

- zero or one CollaborationActivity element,
- zero or more [XML SCHEMA-1] ***wildcard*** elements.

The ***ActionContext*** element also has the following attributes:

- a REQUIRED ***binaryCollaboration*** attribute,
- a REQUIRED businessTransactionActivity attribute,
- a REQUIRED requestOrResponseAction attribute.

8.4.16.1 binaryCollaboration attribute

The REQUIRED ***binaryCollaboration*** attribute is a string that identifies the ***BinaryCollaboration*** for which the parent ***ThisPartyActionBinding*** is defined. If the *Process-Specification* document is defined by the ebXML Business Process Specification Schema[ebBPSS], then the value of the ***binaryCollaboration*** attribute MUST match the value of the ***name*** attribute of the ***BinaryCollaboration*** element as defined in the ebXML Business Process Specification Schema[ebBPSS].

8.4.16.2 businessTransactionActivity attribute

The REQUIRED ***businessTransactionActivity*** attribute is a string that identifies the *Business Transaction* for which the parent ***ThisPartyActionBinding*** is defined. If the *Process-Specification* document is defined by the ebXML Business Process Specification Schema[ebBPSS], the value of the ***businessTransactionActivity*** attribute MUST match the value of the ***name*** attribute of the *BusinessTransactionActivity* element, whose parent is the ***BinaryCollaboration*** referred to by the ***binaryCollaboration*** attribute.

8.4.16.3 requestOrResponseAction attribute

The REQUIRED ***requestOrResponseAction*** attribute is a string that identifies either the *Requesting or Responding Business Activity* for which the parent ***ThisPartyActionBinding*** is defined. For a ***ThisPartyActionBinding*** defined for the request side of a message exchange, if the *Process-Specification* document is defined by the ebXML Business Process Specification Schema [ebBPSS], the value of the ***requestOrResponseAction*** attribute MUST match the value

of the ***name*** attribute of the ***RequestingBusinessActivity*** element corresponding to the *Business Transaction* specified in the ***businessTransactionActivity*** attribute. Similarly, for the response side of a message exchange, the value of the ***requestOrResponseAction*** attribute MUST match the value of the ***name*** attribute of the ***RespondingBusinessActivity*** element corresponding to the *Business Transaction* specified in the ***businessTransactionActivity*** attribute, as defined in the ebXML Business Process Specification Schema[ebBPSS].

8.4.17 CollaborationActivity element

The ***CollaborationActivity*** element supports the ***ActionContext*** element by providing the ability to map any nested ***BinaryCollaboration***s as defined in the ebXML Business Process Specification Schema[ebBPSS] to the ***action*** attribute. The ***CollaborationActivity*** element MUST be present when the ***BinaryCollaboration*** referred to by the ***binaryCollaboration*** attribute has a ***CollaborationActivity*** defined in the business process definition.

An example of the ***CollaborationActivity*** element is:

```
<tp:CollaborationActivity
            tp:name="Credit Check"/>
```

The ***CollaborationActivity*** element has zero or one child ***CollaborationActivity*** element to indicate further nesting of ***BinaryCollaboration****s*.

The ***CollaborationActivity*** element also has one attribute:

- a REQUIRED ***name*** attribute.

8.4.17.1 name attribute

The REQUIRED ***name*** attribute is a string that identifies the ***CollaborationActivity*** included in the ***BinaryCollaboration***. If the *Process-Specification* document is defined by the ebXML Business Process Specification Schema[ebBPSS], the value of the ***name*** attribute MUST match the value of the ***name*** attribute of the ***CollaborationActivity*** within the ***BinaryCollaboration****,* as defined in the ebXML Business Process Specification Schema[ebBPSS].

8.4.18 Certificate element

The ***Certificate*** element defines certificate information for use in this *CPP*. One or more ***Certificate*** elements can be provided for use in the various security functions in the *CPP*. An example of the ***Certificate*** element is:

```
<tp:Certificate tp:certId="CompanyA_SigningCert">
    <ds:KeyInfo>. . .</ds:KeyInfo>
</tp:Certificate>
```

The ***Certificate*** element has a single REQUIRED attribute: ***certId***. The ***Certificate*** element has a single child element: ***ds:KeyInfo***.

The ***ds:KeyInfo*** element may contain a complete chain of certificates, but the leaf certificate is the ***Certificate*** element containing the key used in various asymmetric cryptographic operations. (The leaf certificate will be one that has been issued but has not been used to issue certificates.) If the leaf certificate has been issued by an intermediate certificate authority, the complete chain to the root certificate authority SHOULD be included because it aids in testing certificate validity with respect to a set of trust anchors.

8.4.18.1 certId attribute

The REQUIRED ***certId*** attribute is an [XML] ID that is referred to by a ***CertificateRef*** element elsewhere in the *CPP*. Here is an example of how a ***CertificateRef*** would refer to the ***Certificate*** element shown in the previous section:

```
<tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
```

8.4.18.2 ds:KeyInfo element

The ***ds:KeyInfo*** element defines the certificate information. The content of this element and any sub-elements are defined by the XML Digital Signature specification[XMLDSIG].

NOTE: Software for creation of *CPPs* and *CPAs* MUST recognize the ***ds:KeyInfo*** element and insert the sub-element structure necessary to define the certificate.

8.4.19 SecurityDetails element

The ***SecurityDetails*** element defines a set of ***TrustAnchors*** and an associated ***SecurityPolicy*** for use in this *CPP*. One or more ***SecurityDetails*** elements can be provided for use in the various security functions in the *CPP.* An example of the ***SecurityDetails*** element is:

```
<tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
      <tp:TrustAnchors tp:trustId="MessageTrustAnchors">
            <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
            <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
      </tp:TrustAnchors>
      <tp:SecurityPolicy> ... </tp:SecurityPolicy>
</tp:SecurityDetails>
```

The ***SecurityDetails*** element has zero or one ***TrustAnchors*** element that identifies a set of certificates that are trusted by the *Party*. It also has zero or one ***SecurityPolicy*** element.

The ***SecurityDetails*** element allows agreement to be reached on what root certificates will be used in checking the validity of the other *Party*'s certificates. It can also specify policy regarding operation of the public key infrastructure.

The ***SecurityDetails*** element has one attribute:

- A REQUIRED ***securityId*** attribute.

8.4.19.1 securityId attribute

The REQUIRED ***securityId*** attribute is an [XML] ID that is referred to by an element elsewhere in the *CPP*. Here is an example of how a ***SigningSecurityDetailsRef*** would refer to the ***SecurityDetails*** element shown in the previous section:

```
     <tp:SigningSecurityDetailsRef
tp:securityId="CompanyA_MessageSecurity"/>
```

8.4.20 TrustAnchors element

The ***TrustAnchors*** element contains one or more ***AnchorCertificateRef*** elements, each of which refers to a ***Certificate*** element (under ***PartyInfo***) that represents a certificate trusted by this *Party*. These trusted certificates are used in the process of certificate path validation. If a certificate in question does not "chain" to one of this *Party*'s trust anchors, it is considered invalid.

The TrustAnchors element eventually resolves into XMLDsig ***KeyInfo*** elements. These elements may contain several certificates (a chain), and may refer to those certificates using the ***RetrievalMethod*** element. When there is a chain, the trust anchor is the "leaf" certificate with respect to the "root" issuing certificate authority (CA) certificate. The root CA will be a self-issued and self-signed certificate, and using the Issuer information and perhaps key usage attributes, the leaf certificate ("issued but not issuing" within the chain) can be determined. The chain is included for convenience in that validity checks typically will chain to a "root" CA. Please note that the inclusion of a root CA in a chain does not mean that the root CA is being announced as a trust anchor. It is possible for there to be a PKI policy in which some, but not all, intermediate CAs are trusted. If a root CA were accepted as a trust anchor, all of its intermediate CAs, and all the certificates they issue, would be validated. That might not be what was intended.

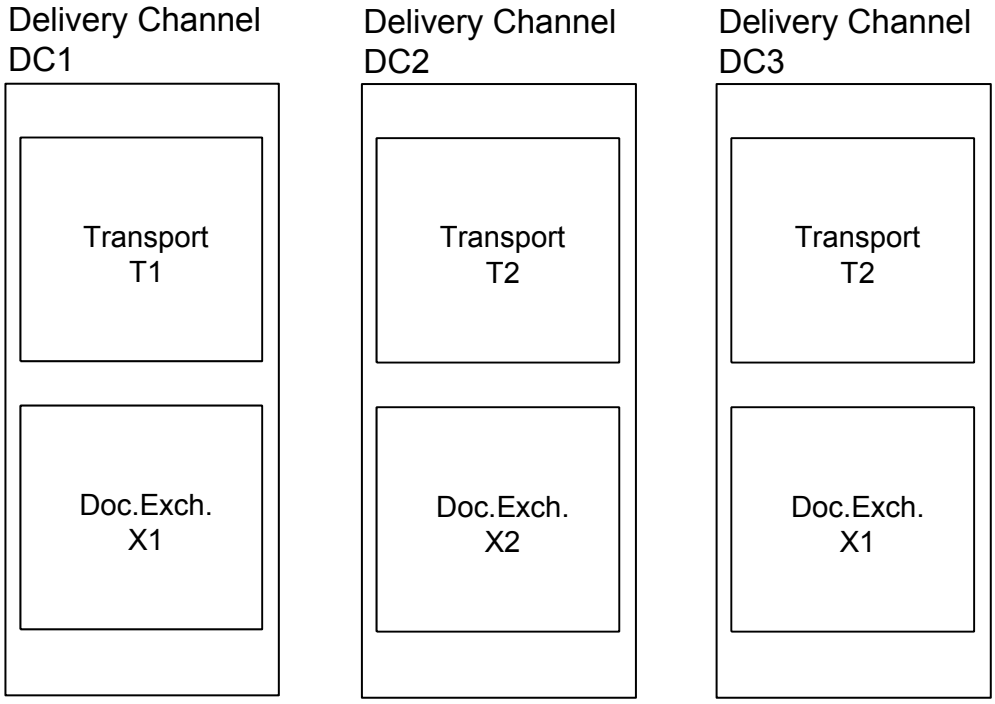
8.4.21 SecurityPolicy element

The ***SecurityPolicy*** element is a placeholder for future apparatus that will enable the *Party* to specify its policy and compliance regarding specific components of its public key infrastructure. For example, it might stipulate revocation checking procedures or constraints related to name, usage, or path length.

8.4.22 DeliveryChannel element

A delivery channel is a combination of a ***Transport*** element and a ***DocExchange*** element that describes the *Party's Message* communication characteristics. The *CPP* SHALL contain one or more ***DeliveryChannel*** elements, one or more ***Transport*** elements, and one or more ***DocExchange*** elements. Each delivery channel SHALL refer to any combination of a ***DocExchange*** element and a ***Transport*** element. The same ***DocExchange*** element or the same ***Transport*** element can be referred to by more than one delivery channel. Two delivery channels can use the same transport protocol and the same document-exchange protocol and differ only in details such as communication addresses or security definitions. Figure 5 illustrates three delivery channels.

Figure 5: Three Delivery Channels



The delivery channels have ID attributes with values "DC1", "DC2", and "DC3". Each delivery channel contains one transport definition and one document-exchange definition. Each transport definition and each document-exchange definition also has an ID attribute whose value is shown in the figure. Note that delivery channel DC3 illustrates that a delivery channel can refer to the same transport definition and document-exchange definition used by other delivery channels but a different combination. In this case delivery channel DC3 is a combination of transport definition T2 (also referred to by delivery channel DC2) and document-exchange definition X1 (also referred to by delivery channel DC1).

Following is the delivery-channel syntax.

```
<tp:DeliveryChannel
    tp:channelId="channel1"
    tp:transportId="transport1"
    tp:docExchangeId="docExchange1"
    <tp:MessagingCharacteristics
        tp:syncReplyMode="none"
        tp:ackRequested="always"
        tp:ackSignatureRequested="always"
        tp:duplicateElimination="always"
        tp:actor="urn:oasis:names:tc:ebxml-msg:actor:nextMSH"/>
</tp:DeliveryChannel>
```

Each ***DeliveryChannel*** element identifies one ***Transport*** element and one ***DocExchange*** element that together make up a single delivery channel definition.

The ***DeliveryChannel*** element has the following attributes:

- a REQUIRED ***channelId*** attribute,
- a REQUIRED ***transportId*** attribute,
- a REQUIRED ***docExchangeId*** attribute.

The ***DeliveryChannel*** element has one REQUIRED child element, ***MessagingCharacteristics.***

-

8.4.22.1 channelId attribute

The ***channelId*** attribute is an [XML] ID attribute that uniquely identifies the ***DeliveryChannel*** element for reference, using IDREF attributes, from other parts of the *CPP* or *CPA*.

8.4.22.2 transportId attribute

The ***transportId*** attribute is an [XML] IDREF that identifies the ***Transport*** element that defines the transport characteristics of the delivery channel. It MUST have a value that is equal to the value of a ***transportId*** attribute of a ***Transport*** element elsewhere within the *CPP* document.

8.4.22.3 docExchangeId attribute

The ***docExchangeId*** attribute is an [XML] IDREF that identifies the ***DocExchange*** element that defines the document-exchange characteristics of the delivery channel. It MUST have a value that is equal to the value of a ***docExchangeId*** attribute of a ***DocExchange*** element elsewhere within the *CPP* document.

8.4.23 MessagingCharacteristics element

The ***MessagingCharacteristics*** element describes the attributes associated with messages delivered over a given delivery channel. The collaborating *Parties* can stipulate that these attributes be fixed for all messages sent through the delivery channel, or they can agree that these attributes be variable on a “per message” basis.

CPP and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel definition (transport and document-exchange) for consistency with these attributes.

The ***MessagingCharacteristics*** element has the following attributes:

- An IMPLIED ***syncReplyMode*** attribute,
- an IMPLIED ***ackRequested*** attribute,
- an IMPLIED ***ackSignatureRequested*** attribute,
- an IMPLIED ***duplicateElimination*** attribute,
- an IMPLIED ***actor*** attribute.

8.4.23.1 syncReplyMode attribute

The ***syncReplyMode*** attribute is an enumeration comprised of the following possible values:

- "mshSignalsOnly"
- "signalsOnly"
- "responseOnly"

- "signalsAndResponse"
- "none"

This attribute, when present, indicates what the sending application expects in a synchronous response (the delivery channel MUST be bound to a synchronous communication protocol such as HTTP when ***syncReplyMode*** is not "none").

The value of "mshSignalsOnly" indicates that the response returned (on the HTTP 200 response in the case of HTTP) will only contain standalone *Message Service Handler (MSH)* level messages like Acknowledgment (for Reliable Messaging) and Error messages. All other application level responses are to be returned asynchronously (using a ***DeliveryChannel*** element determined by the service and action in question).

The value of "signalsOnly" indicates that the response returned (on the HTTP 200 response in the case of HTTP) will only include one or more *Business* signals as defined in the *Process-Specification* document[ebBPSS], plus any piggybacked MSH level signals, but not a *Business*-response *Message*. If the *Process-Specification* calls for the use of a *Business*-response Message, then the latter MUST be returned asynchronously. If the *Business Process* does not call for the use of an *Acceptance Acknowledgment* signal, then the ***Action*** element in the synchronously returned ebXML *Message* MUST be set to "ReceiptAcknowledgment". Otherwise, the ***Action*** element in the synchronously returned ebXML *Message* (which includes both a *Receipt Acknowledgment* signal and an *Acceptance Acknowledgment* signal) MUST be set to "AcceptanceAcknowledgment".

The value of "responseOnly" indicates that any *Business* signals, even if they are indicated in the *Process Specification,* are to be omitted and only the *Business*-response *Message* will be returned synchronously, plus any piggybacked MSH level signals. To be consistent, the ***timeToAcknowledgeReceipt*** and ***timeToAcknowledgeAcceptance*** attributes under the corresponding ***BusinessTransactionCharacteristics*** element SHOULD be set to zero to indicate that these signals are not to be used at all. The ***Action*** element in the synchronously returned ebXML *Message* is determined by the name of the action in the *CPA* that corresponds to the appropriate ***RespondingBusinessActivity*** in the *Business Process*.

The value of "signalsAndResponse" indicates that the application will synchronously return the *Business*-response *Message* in addition to one or more *Business* signals, plus any piggybacked *MSH* level signals. In this case, each signal and response that is bundled into the same ebXML message must appear as a separate MIME part (i.e., be placed in a separate payload container). To be consistent, the ***timeToAcknowledgeReceipt*** and ***timeToPerform*** attributes under the corresponding ***BusinessTransactionCharacteristics*** element SHOULD have identical values. The ***timeToAcknowledgeAcceptance*** attribute, if specified, SHOULD also have the same value as the above two timing attributes. The ***Action*** element in the synchronously returned ebXML *Message* is determined by the name of the action in the *CPA* that corresponds to the appropriate ***RespondingBusinessActivity*** in the *Business Process*.

The *Receipt Acknowledgment* signal for the *Business*-response *Message,* sent from the request

initiator back to the responder, if called for by the *Process-Specification*, MUST also be delivered over the same synchronous connection.

NOTE: For HTTP 1.1 clients and servers, two HTTP requests and replies will have to be sent and received on the same connection. Implementations that implicitly assume that a HTTP connection will be closed after a single synchronous request reply interchange will not be able to support the "signalsAndResponse" synchronous reply mode.

The value of "none", which is the implied default value in the absence of the ***syncReplyMode*** attribute, indicates that neither the *Business*-response *Message* nor any *Business* signal(s) will be returned synchronously. In this case, all *Message Service Handler* level and *Business* level messages will be returned as separate asynchronous messages.

The ebXML *Message* Service's ***SyncReply*** element is included in the SOAP Header whenever the ***syncReplyMode*** attribute has a value other than "none". If the delivery channel identifies a transport protocol that has no synchronous capabilities (such as SMTP), the ***BusinessTransactionCharacteristics*** element SHALL NOT have a ***syncReplyMode*** attribute with a value other than "none".

When the value of the ***syncReplyMode*** attribute is other than "none", a synchronous delivery channel SHALL be used to exchange all messages necessary for conducting a business transaction. If the *Process Specification* calls for the use of non-repudiation of receipt for the response message, then the initiator is expected to return a signed *ReceiptAcknowledgment* signal for the responder's response message.

8.4.23.2 ackRequested attribute

The IMPLIED ***ackRequested*** attribute is an enumeration comprised of the following possible values:

- "always"
- "never"
- "perMessage"

This attribute has the default value "perMessage" meaning whether the ***AckRequested*** element in the SOAP Header is present or absent can be varied on a "per message" basis. If this attribute is set to "always", then every message sent over the delivery channel MUST have an ***AckRequested*** element in the SOAP Header. If this attribute is set to "never", then every message sent over the delivery channel MUST NOT have an ***AckRequested*** element in the SOAP Header.

If the ***ackRequested*** attribute is not set to "never", then the ***ReliableMessaging*** element must be present under the corresponding ***DocExchange*** element to provide the necessary Reliable Messaging parameters.

8.4.23.3 ackSignatureRequested attribute

The IMPLIED ***ackSignatureRequested*** attribute is an enumeration comprised of the following possible values:

- "always"
- "never"
- "perMessage"

This attribute determines how the ***signed*** attribute within the *AckRequested* element in the SOAP Header is to be set. It has the default value "perMessage" meaning that the ***signed*** attribute in the ***AckRequested*** element within the SOAP Header can be set to "true" or "false" on a "per message" basis. If this attribute is set to "always", then every message sent over the delivery channel that has an ***AckRequested*** element in the SOAP Header MUST have its ***signed*** attribute set to "true". If this attribute is set to "never", then every message sent over the delivery channel that has an ***AckRequested*** element in the SOAP Header MUST have its ***signed*** attribute set to "false". If the ***ackRequested*** attribute is set to "never", the setting of the ***ackSignatureRequested*** attribute has no effect.

NOTE: By enabling the use of signed *Acknowledgment* for reliably delivered messages, a weak form of non-repudiation of receipt can be supported. This is considered weaker than the *Receipt Acknowledgment* signal because no schema check can be performed on the payload prior to the return of the *Acknowledgment*. The ***ackSignatureRequested*** attribute can be set independent of the value for the ***isNonRepudiationReceiptRequired*** attribute under the ***BusinessTransactionCharacteristics*** element. Thus, even if the original *Process-Specification* specifies that non-repudiation of receipt is to be performed, the *CPP* and/or *CPA* can override this requirement, set ***isNonRepudiationReceiptRequired*** to "false" and ***ackSignatureRequested*** to "always" and thereby achieve the weak form of non-repudiation of receipt.

8.4.23.4 duplicateElimination attribute

The IMPLIED ***duplicateElimination*** attribute is an enumeration comprised of the following possible values:

- "always"
- "never"
- "perMessage"

This attribute determines whether the ***DuplicateElimination*** element within the ***MessageHeader*** element in the SOAP Header is to be present. It has the default value "perMessage" meaning that the ***DuplicateElimination*** element within the SOAP Header can be present or absent on a "per message" basis. If this attribute is set to "always", then every message sent over the delivery channel MUST have a ***DuplicateElimination*** element in the SOAP Header. If this attribute is set to "never", then every message sent over the delivery channel MUST NOT have a ***DuplicateElimination*** element in the SOAP Header. If the ***duplicateElimination*** attribute is not set to "never", then the ***PersistDuration*** element must be present under the corresponding ***DocExchange*** element to provide the necessary persistent storage parameter.

8.4.23.5 actor attribute

The IMPLIED ***actor*** attribute is an enumeration of the following possible values:

- "urn:oasis:names:tc:ebxml-msg:actor:nextMSH"

- "urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"

This is a URI that will be used as the value for the ***actor*** attribute in the ***AckRequested*** element (see [ebMS]) in case the latter is present in the SOAP Header, as governed by the ***ackRequested*** attribute within the ***MessagingCharacteristics*** element in the *CPA*. If the ***ackRequested*** attribute is set to "never", the setting of the ***actor*** attribute has no effect.

8.4.24 Transport element

The ***Transport*** element defines the *Party's* network communication capabilities. One or more ***Transport*** elements MUST be present in a *CPP*, each of which describes a mechanism the *Party* uses to send messages, a mechanism it uses to receive messages, or both. The following example illustrates the structure of a typical ***Transport*** element:

```
<tp:Transport tp:transportId="transportA1">
    <tp:TransportSender> <!-- 0 or 1 time -->
        <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
        <tp:TransportClientSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">
                SSL
            </tp:TransportSecurityProtocol>
            <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
            <tp:ServerSecurityDetailsRef
                tp:securityId="CompanyA_TransportSecurity"/>
        </tp:TransportClientSecurity>
    </tp:TransportSender>
    <tp:TransportReceiver> <!-- 0 or 1 time -->
        <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
        <tp:Endpoint
            tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler"
            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:TransportServerSecurity>
    </tp:TransportReceiver>
</tp:Transport>
```

The ***Transport*** element consists of zero or one ***TransportSender*** element and zero or one ***TransportReceiver*** element.

A ***Transport*** that contains both ***TransportSender*** and ***TransportReceiver*** elements is said to be *bi-directional* in that it can be used for send and receiving messages. If the *Party* prefers to communicate in synchronous mode (where replies are returned over the same TCP connections messages are sent on; see Section 8.4.23.1), its *CPP* MUST provide a ***ServiceBinding*** that contains ***ActionBindings*** that are bound to a ***DeliveryChannel*** that uses a bi-directional ***Transport***.

A ***Transport*** that contains either a ***TransportSender*** or a ***TransportReceiver*** element, but not both, is said to be *unidirectional*. A unidirectional ***Transport*** can only be used for sending or receiving messages (not both) depending on which element it includes.

A *CPP* contains as many ***Transport*** elements as are needed to fully express the *Party*'s inbound and outbound communication capabilities. If, for example, the *Party* can send and receive messages via HTTP and SMTP, its *CPP* would contain a ***Transport*** element containing its HTTP properties and another ***Transport*** element containing its SMTP properties.

The ***Transport*** element has

- a REQUIRED ***transportId*** attribute.

8.4.24.1 transportId attribute

The REQUIRED ***transportId*** attribute is an [XML] ID that is refers to a ***Transport*** element elsewhere in the *CPP*. Here is an example of a ***DeliveryChannel*** that refers to the ***Transport*** element shown in the previous section:

```
<tp:DeliveryChannel tp:channelId="channelA1"
    tp:transportId="transportA1"
    tp:docExchangeId="docExchangeA1">
    <tp:MessagingCharacteristics . . . />
</tp:DeliveryChannel>
```

8.4.25 TransportSender element

The ***TransportSender*** element contains properties related to the sending side of a ***DeliveryChannel***. Its REQUIRED ***TransportProtocol*** element specifies the transport protocol that will be used for sending messages. The ***AccessAuthentication*** element(s), if present, specifies the type(s) of access authentication supported by the client. The ***TransportClientSecurity*** element, if present, defines the *Party*'s provisions for client-side transport layer security.

The ***TransportSender*** element has no attributes.

8.4.26 TransportProtocol element

The ***TransportProtocol*** element identifies a transport protocol that the *Party* is capable of using to send or receive *Business* data. The IMPLIED ***version*** attribute identifies the specific version of the protocol.

> NOTE: It is the aim of this specification to enable support for any transport capable of carrying MIME content using the vocabulary defined herein.

8.4.27 AccessAuthentication element

The ***AccessAuthentication*** element, if present, indicates the authentication mechanism that MAY be used by a transport server to challenge a client request and by a client to provide

authentication information to a server. For example, [RFC2617] specifies two access authentication schemes for HTTP: "basic" and "digest". A client that supports both would have two ***AccessAuthentication*** elements, as shown below. When multiple schemes are supported, the order in which they are specified in the CPP indicates the order of preference.

```
<tp:TransportSender>
    <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
    <tp:AccessAuthentication>digest</tp:AccessAuthentication>
    <tp:AccessAuthentication>basic</tp:AccessAuthentication>
    <tp:TransportClientSecurity>
        ...
    </tp:TransportClientSecurity>
</tp:TransportSender>
```

NOTE: A *CPA* will contain, for each ***TransportSender*** or ***TransportReceiver***, only the agreed-upon ***AccessAuthentication*** elements.

NOTE: For basic authentication, the userid and password values are configured through means outside of this specification.

8.4.28 TransportClientSecurity element

The ***TransportClientSecurity*** element provides information about this *Party*'s transport client needed by the other *Party*'s transport server to enable a secure connection to be established between the two. It contains a REQUIRED ***TransportSecurityProtocol*** element, zero or one ***ClientCertificateRef*** element, zero or one ***ServerSecurityDetailsRef*** element, and zero or more ***EncryptionAlgorithm*** elements.

In asynchronous messaging mode, the sender will always be a client to the receiver's server. In synchronous messaging mode, the MSH-level reply (and maybe a bundled business signal and/or business response) is sent back over the same connection the initial business message arrived on. In such cases, where the sender is the server and the receiver is the client and the connection already exists, the sender's ***TransportClientSecurity*** and the receiver's ***TransportServerSecurity*** elements SHALL be ignored.

8.4.29 TransportSecurityProtocol element

The ***TransportSecurityProtocol*** element identifies the transport layer security protocol that is supported by the parent ***Transport***. The IMPLIED ***version*** attribute identifies the specific version of the protocol.

For encryption, the protocol is TLS Version 1.0[RFC2246], which uses public-key encryption. Appendix E of the TLS Version 1.0 specification[RFC2246] covers backward compatibility with SSL [SSL].

8.4.30 ClientCertificateRef element

The ***ClientCertificateRef*** element identifies the certificate to be used by the client's transport security module. The REQUIRED IDREF attribute ***certId*** identifies the certificate to be used by

referring to the ***Certificate*** element (under ***PartyInfo***) that has the matching ID attribute value. A TLS-capable HTTP client, for example, uses this certificate to authenticate itself with receiver's secure HTTP server.

The ***ClientCertificateRef*** element, if present, indicates that mutual authentication between client and server (i.e., initiator and responder of the HTTP connection) MUST be performed.

The ***ClientCertificateRef*** element has

- A REQUIRED ***certId*** attribute.
-

8.4.31 ServerSecurityDetailsRef element

The ***ServerSecurityDetailsRef*** element identifies the trust anchors and security policy that this *Party* will apply to the other *Party*'s server authentication certificate.

The ***ServerSecurityDetailsRef*** element has

- A REQUIRED ***securityId*** attribute.
-

8.4.32 Encryption Algorithm

Zero or more ***EncryptionAlgorithm*** elements may be included under the ***TransportClientSecurity*** or ***TransportServerSecurity*** element. Multiple elements are of more use in a CPP context, to announce capabilities or preferences; normally, a CPA will contain the agreed upon context. When zero or more than one element is present in a CPA, the *Parties* agree to allow the automatic negotiation capability of the ***TransportSecurityProtocol*** element to determine the actual algorithm used.

The elements' ordering will reflect the preference for algorithms. A primary reason for including this element is to permit use of the ***minimumStrength*** attribute; a large value for this attribute can indicate that high encryption strength is desired or has been agreed upon for the ***TransportSecurityProtocol***.

See section 8.4.50 for the full description of this element.

For SSL and TLS, it is customary to specify cipher suite values as text values for the ***EncryptionAlgorithm*** element. These values include, but are not limited to:

- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA,
- TLS_RSA_WITH_3DES_EDE_CBC_SHA,
- SSL_RSA_WITH_3DES_EDE_CBC_SHA,
- SSL_RSA_WITH_RC4_128_MD5,
- SSL_RSA_WITH_RC4_128_SHA,
- SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA,
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA.

Consult the original specifications for enumerations and discussions of these values.

8.4.33 TransportReceiver element

The ***TransportReceiver*** element contains properties related to the receiving side of a ***DeliveryChannel***. Its REQUIRED ***TransportProtocol*** element specifies the transport protocol that will be used for receiving messages. One or more REQUIRED ***Endpoint*** elements specify logical addresses where messages can be received. The ***AccessAuthentication*** element(s), if present, indicates the type(s) of access authentication supported by the server. Zero or one ***TransportServerSecurity*** element defines the *Party*'s provisions for server-side transport layer security.

The ***TransportReceiver*** element has no attributes.

8.4.34 Endpoint element

One or more ***Endpoint*** elements SHALL be provided for each ***TransportReceiver*** element. Each ***Endpoint*** specifies a logical address and an indication of what kinds of messages can be received at that location.

Each ***Endpoint*** has the following attributes:

- a REQUIRED ***uri*** attribute,
- an IMPLIED ***type*** attribute.
-

8.4.34.1 uri attribute

The REQUIRED ***uri*** attribute specifies a URI identifying the address of a resource. The value of the ***uri*** attribute SHALL conform to the syntax for expressing URIs as defined in [RFC2396].

8.4.34.2 type attribute

The ***type*** attribute identifies the purpose of this endpoint. The value of ***type*** is an enumeration; permissible values are "login", "request", "response", "error", and "allPurpose". There can be, at most, one of each. If the ***type*** attribute is omitted, its value defaults to "allPurpose". The "login" endpoint is used for the address for the initial *Message* between the two *Parties.* The "request" and "response" endpoints are used for request and response *Messages*, respectively. To enable error *Messages* to be received, each ***Transport*** element SHALL contain at least one endpoint of type "error", "response", or "allPurpose".

The types of ***Endpoint*** element within a ***TransportReceiver*** element MUST not be overlapping. Thus, it would be erroneous to include both an "allPurpose" ***Endpoint*** element along with another ***Endpoint*** element of any type.

8.4.35 TransportServerSecurity element

The ***TransportServerSecurity*** element provides information about this *Party*'s transport server needed by the other *Party*'s transport client to enable a secure connection to be established

between the two. It contains a REQUIRED ***TransportSecurityProtocol*** element, a REQUIRED ***ServerCertificateRef*** element, zero or one ***ClientSecurityDetailsRef*** element, and zero or more ***EncryptionAlgorithm*** elements. See Section 8.4.32 for a description of the ***EncryptionAlgorithm*** element.

NOTE: See the note in Section 8.4.27 regarding the relevance of the ***TransportServerSecurity*** element when synchronous replies are in use.

8.4.36 ServerCertificateRef element

The ***ServerCertificateRef*** element, if present, identifies the certificate to be used by the server's transport security module. The REQUIRED IDREF attribute ***certId*** identifies the certificate to be used by referring to the ***Certificate*** element (under ***PartyInfo***) that has the matching ID attribute value. A TLS-enabled HTTP server, for example, uses this certificate to authenticate itself with the sender's TLS client.

The ***ServerCertificateRef*** element MUST be present if the transport security protocol uses certificates. It MAY be omitted otherwise (e.g. if authentication is by password).

The ***ServerCertificateRef*** element has

- A REQUIRED ***certId*** attribute.
-

8.4.37 ClientSecurityDetailsRef element

The ***ClientSecurityDetailsRef*** element, if present, identifies the trust anchors and security policy that this *Party* will apply to the other *Party*'s client authentication certificate.

The ***ClientSecurityDetailsRef*** element has

- A REQUIRED ***securityId*** attribute.

8.4.38 Transport protocols

In the following sections, we discuss the specific details of each supported transport protocol.

8.4.38.1 HTTP

HTTP is Hypertext Transfer Protocol[HTTP]. For HTTP, the endpoint is a URI that SHALL conform to [RFC2396]. Depending on the application, there MAY be one or more endpoints, whose use is determined by the application.

Following is an example of an HTTP endpoint:

```
<tp:Endpoint tp:uri="http://example.com/servlet/ebxmlhandler"
    tp:type="request"/>
```

The "request" and "response" endpoints can be dynamically overridden for a particular request or asynchronous response by application-specified URIs in *Business* documents exchanged under the *CPA*.

For a synchronous response, the "response" endpoint is ignored if present. A synchronous response is always returned on the existing connection, i.e. to the URI that is identified as the source of the connection.

8.4.38.2 SMTP

SMTP is Simple Mail Transfer Protocol[SMTP]. For use with this standard, Multipurpose Internet Mail Extensions[MIME] MUST be supported. For SMTP, the communication address is the fully qualified mail address of the destination *Party* as defined by [RFC2822]. Following is an example of an SMTP endpoint:

```
<tp:Endpoint tp:uri="mailto:ebxmlhandler@example.com"
    tp:type="request"/>
```

NOTE: The SMTP Mail Transfer Agent (MTA) can encode binary data when the receiving MTA does not support binary transfer. In general, SMTP transfer may involve coding and recoding of Content-Transfer-Encodings as a message moves along a sequence of MTAs. Such changes can in some circumstances invalidate some kinds of signatures even though no malicious actions or transmission errors have occurred.

NOTE: SMTP by itself (without any authentication or encryption) is subject to denial of service and masquerading by unknown *Parties*. It is strongly suggested that those *Parties* who choose SMTP as their transport layer also choose a suitable means of encryption and authentication either in the document-exchange layer or in the transport layer such as [S/MIME].

NOTE: SMTP is an asynchronous protocol that does not guarantee a particular quality of service. A transport-layer acknowledgment (i.e. an SMTP acknowledgment) to the receipt of a mail *Message* constitutes an assertion on the part of the SMTP server that it knows how to deliver the mail *Message* and will attempt to do so at some point in the future. However, the *Message* is not hardened and might never be delivered to the recipient. Furthermore, the sender will see a transport-layer acknowledgment only from the nearest node. If the *Message* passes through intermediate nodes, SMTP does not provide an end-to-end acknowledgment. Therefore receipt of an SMTP acknowledgement does not guarantee that the *Message* will be delivered to the application and failure to receive an SMTP acknowledgment is not evidence that the *Message* was not delivered. It is RECOMMENDED that the reliable-messaging protocol in the ebXML *Message* Service be used with SMTP.

8.4.38.3 FTP

FTP is File Transfer Protocol[RFC959].

Each *Party* sends a *Message* using FTP PUT. The endpoint specifies the user id and input

directory path (for PUTs to this *Party*). An example of an FTP endpoint is:

```
<tp:Endpoint uri="ftp://userid@server.foo.com"
    tp:type="request"/>
```

Since FTP needs to be compatible across all implementations, the FTP for ebXML will use the minimum sets of commands and parameters available for FTP as specified in [RFC959], Section 5.1, and modified in [RFC1123], Section 4.1.2.13. The mode SHALL be stream only and the type MUST be ASCII Non-print (AN), Image (I) (binary), or Local 8 (L 8) (binary between 8-bit machines and machines with 36 bit words – for an 8-bit machine Local 8 is the same as Image).

Stream mode closes the data connection upon end of file. The server side FTP MUST set control to "PASV" before each transfer command to obtain a unique port pair if there are multiple third party sessions.

NOTE: [RFC 959] states that User-FTP SHOULD send a PORT command to assign a non-default data port before each transfer command is issued to allow multiple transfers during a single FTP because of the long delay after a TCP connection is closed until its socket pair can be reused.

NOTE: The format of the 227 reply to a PASV command is not well standardized and an FTP client might assume that the parentheses indicated in [RFC959] will be present when in some cases they are not. If the User-FTP program doesn't scan the reply for the first digit of host and port numbers, the result will be that the User-FTP might point at the wrong host. In the response, the h1, h2, h3, h4 is the IP address of the server host and the p1, p2 is a non-default data transfer port that PASV has assigned.

NOTE: As a recommendation for firewall transparency, [RFC1579] proposes that the client sends a PASV command, allowing the server to do a passive TCP open on some random port, and inform the client of the port number. The client can then do an active open to establish the connection.

NOTE: Since STREAM mode closes the data connection upon end of file, the receiving FTP might assume abnormal disconnect if a 226 or 250 control code hasn't been received from the sending machine.

NOTE: [RFC1579] also makes the observation that it might be worthwhile to enhance the FTP protocol to have the client send a new command APSV (all passive) at startup that would allow a server that implements this option to always perform a passive open. A new reply code 151 would be issued in response to all file transfer requests not preceded by a PORT or PASV command; this *Message* would contain the port number to use for that transfer. A PORT command could still be sent to a server that had previously received APSV; that would override the default behavior for the next transfer operation, thus permitting third-party transfers.

8.4.39 DocExchange Element

The ***DocExchange*** element provides information that the *Parties* MUST agree on regarding exchange of documents between them. This information includes the messaging service properties (e.g. ebXML *Message* Service[ebMS]).

Following is the structure of the ***DocExchange*** element of the *CPP*. Subsequent sections describe each child element in greater detail.

```
<tp:DocExchange tp:docExchangeId="docExchangeB1">
    <tp:ebXMLSenderBinding tp:version="2.0">    <!-- 0 or 1 -->
        <tp:ReliableMessaging>                  <!-- 0 or 1 -->
        . . .
        </tp:ReliableMessaging>
        <tp:PersistDuration>                    <!-- 0 or 1 -->
        . . .
        </tp:PersistDuration>
        <tp:SenderNonRepudiation>               <!-- 0 or 1 -->
        . . .
        </tp:SenderNonRepudiation>
        <tp:SenderDigitalEnvelope>              <!-- 0 or 1 -->
        . . .
        </tp:SenderDigitalEnvelope>
        <tp:NamespaceSupported>                 <!-- 0 or more -->
        . . .
        </tp:NamespaceSupported>
    </tp:ebXMLSenderBinding>
    <tp:ebXMLReceiverBinding tp:version="2.0">  <!-- 0 or 1 -->
        <tp:ReliableMessaging>                  <!-- 0 or 1 -->
        . . .
        </tp:ReliableMessaging>
        <tp:PersistDuration>                    <!-- 0 or 1 -->
        . . .
        </tp:PersistDuration>
        <tp:ReceiverNonRepudiation>             <!-- 0 or 1 -->
        . . .
        </tp:ReceiverNonRepudiation>
        <tp:ReceiverDigitalEnvelope>            <!-- 0 or 1 -->
        . . .
        </tp:ReceiverDigitalEnvelope>
        <tp:NamespaceSupported>                 <!-- 0 or more -->
        . . .
        </tp:NamespaceSupported>
    </tp:ebXMLReceiverBinding>
</tp:DocExchange>
```

The ***DocExchange*** element is comprised of zero or one ***ebXMLSenderBinding*** child element and zero or one ***ebXMLReceiverBinding*** child element**.** It MUST have at least one child element. *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL verify the presence of a child element.

NOTE: The document-exchange section can be extended to messaging services other than the ebXML *Message* service by adding additional ***xxxSenderBinding*** and ***xxxReceiverBinding*** elements and their child elements that describe the other services, where ***xxx*** is replaced by the name of the additional binding. An example is ***XMLPSenderBinding/XMLPReceiverBinding***, which might define support for the future

XML Protocol specification.

8.4.39.1 docExchangeId attribute

The ***DocExchange*** element has a single REQUIRED ***docExchangeId*** attribute that is an [XML] ID that provides a unique identifier that can be referenced from elsewhere within the *CPP* document.

8.4.40 ebXMLSenderBinding element

The ***ebXMLSenderBinding*** element describes properties related to sending messages with the ebXML *Message* Service[ebMS]. The ***ebXMLSenderBinding*** element is comprised of the following child elements:

- zero or one ***ReliableMessaging*** element which specifies the characteristics of reliable messaging,
- zero or one ***PersistDuration*** element which specifies the duration for which certain messages have to be stored persistently for the purpose of duplicate elimination,
- zero or one ***SenderNonRepudiation*** element which specifies the sender's requirements and certificate for message signing,
- zero or one ***SenderDigitalEnvelope*** element which specifies the sender's requirements for encryption by the digital-envelope[DIGENV] method,
- zero or more ***NamespaceSupported*** elements that identify any namespace extensions supported by the messaging service implementation.

The ***ebXMLSenderBinding*** element has one attribute:

- a REQUIRED ***version*** attribute.

NOTE: A CPA could be valid even when omitting all children under ***ebXMLSenderBinding.***

8.4.40.1 version attribute

The REQUIRED ***version*** attribute identifies the version of the ebXML *Message* Service specification being used.

8.4.41 ReliableMessaging element

The ***ReliableMessaging*** element specifies the properties of reliable ebXML *Message* exchange. The default that applies if the ***ReliableMessaging*** element is omitted is "BestEffort". The following is the element structure:

```
<tp:ReliableMessaging>
    <tp:Retries>5</tp:Retries>
    <tp:RetryInterval>PT2H</tp:RetryInterval>
    <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
</tp:ReliableMessaging>
```

Semantics of reliable messaging are explained in the ebXML *Message* Service specification[ebMS] chapter on Reliable Messaging Combinations.

The ***ReliableMessaging*** element is comprised of the following child elements.

- zero or one ***Retries*** element,
- zero or one ***RetryInterval*** element,
- a REQUIRED MessageOrderSemantics element.

8.4.41.1 Retries and RetryInterval elements

The ***Retries*** and ***RetryInterval*** elements specify the permitted number of retries and the interval, expressed as an XML Schema[XMLSCHEMA-2] duration, between retries of sending a reliably delivered *Message* following a timeout waiting for the *Acknowledgment*. The purpose of the ***RetryInterval*** element is to improve the likelihood of success on retry by deferring the retry until any temporary conditions that caused the error might be corrected. The RetryInterval applies to the time between sending of the original message and the first retry, as well as the time between all subsequent retries.

The ***Retries*** and ***RetryInterval*** elements MUST either be included together or be omitted together. If they are omitted, the values of the corresponding quantities (number of retries and retry interval) are a local matter at each *Party.*

8.4.41.2 MessageOrderSemantics element

The ***MessageOrderSemantics*** element is an enumeration comprised of the following possible values:

- "Guaranteed"
- "NotGuaranteed"

The presence of a ***MessageOrderSemantics*** element in the SOAP Header for ebXML messages determines if the ordering of messages sent from the *From Party* needs to be preserved so that the *To Party* receives those messages in the order in which they were sent. If the ***MessageOrderSemantics*** element is set to "Guaranteed", then the ebXML message MUST contain a ***MessageOrder*** element in the SOAP Header. If the ***MessageOrderSemantics*** element is set to "NotGuaranteed", then the ebXML message MUST NOT contain a ***MessageOrder*** element in the SOAP Header. Guaranteed message ordering implies the use of duplicate elimination. Therefore, the PersistDuration element MUST also appear if MessageOrderSemantics is set to "Guaranteed".

8.4.42 PersistDuration element

The value of the ***PersistDuration*** element is the minimum length of time, expressed as an XML Schema[XMLSCHEMA-2] duration, that data from a *Message* that is sent reliably is kept in *Persistent Storage* by an ebXML *Message*-Service implementation that receives that *Message* to facilitate the elimination of duplicates. This duration also applies to response messages that are kept persistently to allow automatic replies to duplicate messages without their repeated processing by the application. For rules that govern the ***PersistDuration*** element, refer to Sections 8.4.23.4 and 8.4.41.2.

8.4.43 SenderNonRepudiation element

The ***SenderNonRepudiation element*** conveys the message sender's requirements and certificate for non-repudiation. Non-repudiation both proves who sent a *Message* and prevents later repudiation of the contents of the *Message*. Non-repudiation is based on signing the *Message* using XML Digital Signature[XMLDSIG]. The element structure is as follows:

```
<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>
```

If the ***SenderNonRepudiation*** element is omitted, the *Messages* are not digitally signed.

The ***SenderNonRepudiation*** element is comprised of the following child elements:

- a REQUIRED NonRepudiationProtocol element,
- a REQUIRED ***HashFunction*** (e.g. SHA1, MD5) element,
- a REQUIRED ***SignatureAlgorithm*** element,
- a REQUIRED SigningCertificateRef element

8.4.44 NonRepudiationProtocol element

The REQUIRED ***NonRepudiationProtocol*** element identifies the technology that will be used to digitally sign a *Message*. It has a single IMPLIED ***version*** attribute whose value is a string that identifies the version of the specified technology.

8.4.45 HashFunction element

The REQUIRED ***HashFunction*** element identifies the algorithm that is used to compute the digest of the *Message* being signed.

8.4.46 SignatureAlgorithm element

The REQUIRED ***SignatureAlgorithm*** element identifies the algorithm that is used to compute the value of the digital signature. Expected values include: RSA-MD5, RSA-SHA1, DSA-MD5, DSA-SHA1, SHA1withRSA, MD5withRSA, and so on.

NOTE: Implementations should be prepared for values in upper and/or lower case and with varying usage of hyphens and conjunctions.

The ***SignatureAlgorithm*** element has three attributes:

- an IMPLIED ***oid*** attribute,
- an IMPLIED ***w3c*** attribute,
- an IMPLIED ***enumeratedType*** attribute.

8.4.46.1 oid attribute

The ***oid*** attribute serves as a way to supply an object identifier for the signature algorithm. The formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D of X.208 (http://www.itu.int/POD/). Commonly used values (in the IETF dotted integer format) for signature algorithms include:

- 1.2.840.113549.1.1.4 - MD5 with RSA encryption,
- 1.2.840.113549.1.1.5 - SHA-1 with RSA Encryption.

8.4.46.2 w3c attribute

The ***w3c*** attribute serves as a way to supply an object identifier for the signature algorithm. The definitions of these values are found in the [XMLDSIG] or [XMLENC] specifications. Expected values for signature algorithms include:

- http://www.w3.org/2000/09/xmldsig#dsa-sha1,
- http://www.w3.org/2000/09/xmldsig#rsa-sha1.

8.4.46.3 enumeratedType attribute

The ***enumeratedType*** attribute specifies a different way of interpreting the text value of the ***SignatureAlgorithm*** element**.** This attribute is for identifying future signature algorithm identification schemes and formats.

8.4.47 SigningCertificateRef element

The REQUIRED ***SigningCertificateRef*** element identifies the certificate the sender uses for signing messages. Its REQUIRED IDREF attribute, ***certId*** refers to the ***Certificate*** element (under ***PartyInfo***) that has the matching ID attribute value.

8.4.48 SenderDigitalEnvelope element

The ***SenderDigitalEnvelope*** element provides the sender's requirements for message encryption using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in which the *Message* is encrypted by symmetric encryption (shared secret key) and the secret key is sent to the *Message* recipient encrypted with the recipient's public key. The element structure is:

```
<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>
```

The ***SenderDigitalEnvelope*** element contains

- a REQUIRED DigitalEnvelopeProtocol element,
- a REQUIRED ***EncryptionAlgorithm*** element
- zero or one EncryptionSecurityDetailsRef element.
-

8.4.49 DigitalEnvelopeProtocol element

The REQUIRED ***DigitalEnvelopeProtocol*** element identifies the message encryption protocol to be used. The REQUIRED ***version*** attribute identifies the version of the protocol.

8.4.50 EncryptionAlgorithm element

The REQUIRED ***EncryptionAlgorithm*** element identifies the encryption algorithm to be used. See also Section 8.4.32.

The ***EncryptionAlgorithm*** element has four attributes:

- an IMPLIED ***minimumStrength*** attribute,
- an IMPLIED ***oid*** attribute,
- an IMPLIED ***w3c*** attribute,
- an IMPLIED ***enumeratedType*** attribute.
-

8.4.50.1 minimumStrength attribute

The ***minimumStrength*** attribute describes the effective strength the encryption algorithm MUST provide in terms of "effective" or random bits. This value is less than the key length in bits when check bits are used in the key. So, for example, the 8 check bits of a 64-bit DES key would not be included in the count, and to require a minimum strength the same as that supplied by DES would be reported by setting ***minimumStrength*** to 56.

8.4.50.2 oid attribute

The ***oid*** attribute serves as a way to supply an object identifier for the encryption algorithm. The formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D of X.208 (http://www.itu.int/POD/). Commonly used values (in the IETF dotted integer format) for encryption algorithms include:

- 1.2.840.113549.3.2 (RC2-CBC),1.2.840.113549.3.4 (RC4 Encryption Algorithm),
- 1.2.840.113549.3.7 (DES-EDE3-CBC), 1.2.840.113549.3.9 (RC5 CBC Pad),
- 1.2.840.113549.3.10 (DES CDMF), 1.2.840, 1.3.14.3.2.7 (DES-CBC).

8.4.50.3 w3c attribute

The ***w3c*** attribute serves as a way to supply an object identifier for the encryption algorithm. The definitions of these values are in the [XMLENC] specification. Expected values include:

- http://www.w3.org/2001/04/xmlenc#3des-cbc,
- http://www.w3.org/2001/04/xmlenc#aes128-cbc,
- http://www.w3.org/2001/04/xmlenc#aes256-cbc.

8.4.50.4 enumeratedTypeAttribute

The ***enumeratedType*** attribute specifies a way of interpreting the text value of the ***EncryptionAlgorithm*** element. This attribute is for identifying future algorithm identification schemes and formats.

8.4.51 EncryptionSecurityDetailsRef element

The ***EncryptionSecurityDetailsRef*** element identifies the trust anchors and security policy that this (sending) *Party* will apply to the other (receiving) *Party*'s encryption certificate. Its REQUIRED IDREF attribute, ***securityId,*** refers to the ***SecurityDetails*** element (under ***PartyInfo***) that has the matching ID attribute value.

8.4.52 NamespaceSupported element

The ***NamespaceSupported*** element may be included zero or more times. Each occurrence of the ***NamespaceSupported*** element identifies one namespace supported by the messaging service implementation. It has a REQUIRED ***location*** attribute and an IMPLIED ***version*** attribute. The ***location*** attribute supplies a URI for retrieval of the schema associated with the namespace. The ***version*** attribute provides a version value, when one exists, for the namespace. While the ***NamespaceSupported*** element can be used to list the namespaces that could be expected to be used during document exchange, the motivation is primarily for extensions, version variants, and other enhancements that might not be expected, or have only recently emerged into use.

For example, support for Security Assertion Markup Language[SAML] would be defined as follows:

```
<tp:NamespaceSupported
tp:location="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-
assertion-27.xsd" tp:version="1.0">
http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-27.xsd</tp:NamespaceSupported>
```

In addition, the ***NamespaceSupported*** element can be used to identify the namespaces associated with the message body parts (see Section 8.5), and especially when these namespaces are not implicitly indicated through parts of the ***ProcessSpecification*** or when they indicate extensions of namespaces for payload body parts.

8.4.53 ebXMLReceiverBinding element

The ***ebXMLReceiverBinding*** element describes properties related to receiving messages with the ebXML *Message* Service[ebMS]. The ***ebXMLReceiverBinding*** element is comprised of the following child elements:

- zero or one ***ReliableMessaging*** element (see Section 8.4.41),
- zero or one ***ReceiverNonRepudiation*** element which specifies the receiver's requirements for message signing,

- zero or one ***ReceiverDigitalEnvelope*** element which specifies the receiver's requirements and certificate for encryption by the digital-envelope[DIGENV] method,
- zero or more ***NamespaceSupported*** elements (see Section 8.4.52).

The ***ebXMLReceiverBinding*** element has one attribute:

- a REQUIRED ***version*** attribute (see Section 8.4.40.1)

NOTE: A CPA could be valid even when omitting all children under ***ebXMLReceiverBinding.***

8.4.54 ReceiverNonRepudiation element

The ***ReceiverNonRepudiation element*** conveys the message receiver's requirements for non-repudiation. Non-repudiation both proves who sent a *Message* and prevents later repudiation of the contents of the *Message*. Non-repudiation is based on signing the *Message* using XML Digital Signature[XMLDSIG]. The element structure is as follows:

```
<tp:ReceiverNonRepudiation>
    <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:SigningSecurityDetailsRef
tp:securityId="CompanyA_MessageSecurity"/>
</tp:ReceiverNonRepudiation>
```

If the ***ReceiverNonRepudiation*** element is omitted, the *Messages* are not digitally signed.

The ***ReceiverNonRepudiation*** element is comprised of the following child elements:

- a REQUIRED ***NonRepudiationProtocol*** element (see Section 8.4.44),
- a REQUIRED ***HashFunction*** (e.g. SHA1, MD5) element (see Section 8.4.45),
- a REQUIRED ***SignatureAlgorithm*** element (see Section 8.4.46),
- zero or one SigningSecurityDetailsRef element
-

8.4.55 SigningSecurityDetailsRef element

The ***SigningSecurityDetailsRef*** element identifies the trust anchors and security policy that this (receiving) *Party* will apply to the other (sending) *Party*'s signing certificate. Its REQUIRED IDREF attribute, ***securityId,*** refers to the ***SecurityDetails*** element (under ***PartyInfo***) that has the matching ID attribute value.

8.4.56 ReceiverDigitalEnvelope element

The ***ReceiverDigitalEnvelope*** element provides the receiver's requirements for message encryption using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in

which the *Message* is encrypted by symmetric encryption (shared secret key) and the secret key is sent to the *Message* recipient encrypted with the recipient's public key. The element structure is:

```
<tp:ReceiverDigitalEnvelope>
    <tp:DigitalEnvelopeProtocol tp:version="2.0">
        S/MIME
    </tp:DigitalEnvelopeProtocol>
    <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
    <tp:EncryptionCertificateRef
        tp:certId="CompanyA_EncryptionCert"/>
</tp:ReceiverDigitalEnvelope>
```

The ***ReceiverDigitalEnvelope*** element contains

- a REQUIRED ***DigitalEnvelopeProtocol*** element (see Section 8.4.49),
- a REQUIRED ***EncryptionAlgorithm*** element (see Section 8.4.50),
- a REQUIRED EncryptionCertificateRef element.

8.4.57 EncryptionCertificateRef element

The REQUIRED ***EncryptionCertificateRef*** element identifies the certificate the sender uses for encrypting messages. Its REQUIRED IDREF attribute, ***certId*** refers to the ***Certificate*** element (under ***PartyInfo***) that has the matching ID attribute value.
.

8.4.58 OverrideMshActionBinding element

The ***OverrideMshActionBinding*** element can occur zero or more times. It has two REQUIRED attributes. The ***action*** attribute identifies the *Message Service Handler* level action whose delivery is not to use the default ***DeliveryChannel*** for *Message Service Handler* actions. The channelId attribute specifies the ***DeliveryChannel*** to be used instead.

8.5 SimplePart element

The ***SimplePart*** element provides a repeatable list of the constituent parts, primarily identified by the MIME content-type value. The ***SimplePart*** element has two REQUIRED attributes: ***id*** and ***mimetype***. The ***id*** attribute, of type ID, provides the value that will be used later to reference this *Message* part when specifying how the parts are packaged into composites, if composite packaging is present. The ***mimetype*** attribute can provide actual values of content-type for the simple *Message* part being specified. The attribute's values may also make use of an asterisk wildcard, "*", to indicate either an arbitrary top-level type, an arbitrary sub-type, or a completely arbitrary type, "*/*". SimpleParts with wildcards in types can be used in indicating more open packaging processing capabilities.

SimplePart has an IMPLIED ***mimparameters*** attribute, whose use is described in section 8.6.2. ***SimplePart*** also has an IMPLIED ***xlink:role*** attribute which identifies some resource that describes the mime part or its purpose; see Appendix F for a discussion of the use of this value within [ebMS]. If present, then it SHALL have a value that is a valid URI in accordance with the [XLINK] specification.

The following are examples of ***SimplePart*** elements:

```
<tp:SimplePart tp:id="I001" tp:mimetype="text/xml"/>
<tp:SimplePart tp:id="I002" tp:mimetype="application/xml"/>
<tp:SimplePart tp:id="I002" tp:mimetype="*/xml"/>
```

The ***SimplePart*** element can have zero or more ***NamespaceSupported*** elements. Each of these identifies any namespace supported for the XML that is packaged in the parent simple body part.

The context of ***Packaging*** can very easily render it pointless to list all the namespaces used in a ***SimplePart***. For example, when defining the ***SimplePart*** for a SOAP envelope, as part of an ebXML Message, it is not necessary to list all the namespaces. If, however, any unusual extensions, new versions, or unusual security extensions are present, it is useful to announce these departures explicitly in the packaging. It is not, however, incorrect to list all namespaces used in a ***SimplePart***, even where these namespaces have been mandated by a given messaging protocol. By convention, when a full listing of namespaces is supplied within a ***SimplePart*** element, the first ***NamespaceSupported*** element identifies the schema for the ***SimplePart*** while subsequent ***NamespaceSupported*** elements represent namespaces that are imported by that schema. Any additional ***NamespaceSupported*** elements indicate extensions.

> NOTE: The explicit identification of imported namespaces is discretionary. Thus, the CPP and CPA examples in Appendix A and Appendix B explicitly identify the ebXML Messaging Service namespace but omit the SOAP envelope and XML Digital Signature namespaces that are imported into the schema for the ebXML Messaging Service namespace.

The same ***SimplePart*** element can be referenced from (i.e., reused in) multiple ***Packaging*** elements.

8.6 Packaging element

The subtree of the ***Packaging*** element provides specific information about how the *Message Header* and payload constituent(s) are packaged for transmittal over the transport, including the crucial information about what document-level security packaging is used and the way in which security features have been applied. Typically the subtree under the ***Packaging*** element indicates the specific way in which constituent parts of the *Message* are organized. MIME processing capabilities are typically the capabilities or agreements described in this subtree. The ***Packaging*** element provides information about MIME content types, XML namespaces, security parameters, and MIME structure of the data that is exchanged between *Parties*.

The following is an example of a ***Packaging*** element which references the example ***SimplePart*** elements given in Section 8.5:

```
<!-- Simple ebXML S/MIME Packaging for application-based payload
     encryption -->
<tp:Packaging>
```

```
            <tp:ProcessingCapabilities tp:generate="true" tp:parse="true"/>
            <tp:CompositeList>
                <tp:Encapsulation
                    <!-- I002 is the payload being encrypted -->
                    tp:id="I003"
                    tp:mimetype="application/pkcs7-mime"
                    tp:mimeparameters="smime-type=&quot;enveloped-data&quot">
                    <Constituent tp:idref="I002"/>
                </tp:Encapsulation>
                <tp:Composite tp:id="I004"
                    <!-- I001 is the SOAP envelope. The ebXML message is made
                         up of the SOAP envelope and the encrypted payload. -->
                    tp:mimetype="multipart/related"
                    tp:mimeparameters="type=&quot;text/xml&quot;
        version=&quot;1.0&quot">
                    <tp:Constituent tp:idref="I001"/>
                    <tp:Constituent tp:idref="I003"/>
                </tp:Composite>
            </tp:CompositeList>
        </tp:Packaging>
```

The ***Packaging*** element has one attribute; the REQUIRED ***id*** attribute, with type ID. It is referred to in the ***ThisPartyActionBinding*** element, by using the IDREF attribute, ***packageId.***

The child elements of the ***Packaging*** element are ***ProcessingCapabilities*** and ***CompositeList.*** This set of elements can appear one or more times as a child of each ***Packaging*** element.

8.6.1 ProcessingCapabilities element

The ***ProcessingCapabilities*** element has two REQUIRED attributes with Boolean values of either "true" or "false". The attributes are ***parse*** and ***generate***. Normally, these attributes will both have values of "true" to indicate that the packaging constructs specified in the other child elements can be both produced as well as processed at the software *Message* service layer. At least one of the ***generate*** or ***parse*** attributes MUST be true.

8.6.2 CompositeList element

The final child element of ***Packaging*** is ***CompositeList***, which is a container for the specific way in which the simple parts are combined into groups (MIME multiparts) or encapsulated within security-related MIME content-types. The ***CompositeList*** element SHALL be omitted from ***Packaging*** when no security encapsulations or composite multiparts are used. When the ***CompositeList*** element is present, the content model for the ***CompositeList*** element is a repeatable sequence of choices of ***Composite*** or ***Encapsulation*** elements. The ***Composite*** and ***Encapsulation*** elements can appear intermixed as desired. The sequence in which the choices are presented is important because, given the recursive character of MIME packaging, composites or encapsulations can include previously mentioned composites (or rarely, encapsulations) in addition to the *Message* parts characterized within the ***SimplePart*** subtree. Therefore, the "top-level" packaging will be described last in the sequence.

The ***Composite*** element has the following attributes:

- a REQUIRED ***mimetype*** attribute,

- a REQUIRED ***id*** attribute,
- an IMPLIED ***mimeparameters*** attribute.

The ***mimetype*** attribute provides the value of the MIME content-type for this *Message* part, and this will be some MIME composite type, such as "multipart/related" or "multipart/signed". The ***id*** attribute, type ID, provides a way to refer to this composite if it needs to be mentioned as a constituent of some later element in the sequence. The ***mimeparameters*** attribute provides the values of any significant MIME parameter (such as "type=application/xml") that is needed to understand the processing demands of the content-type.

The ***Composite*** element has one child element, ***Constituent.***

The ***Constituent*** element has one REQUIRED attribute, ***idref*** of type IDREF, an IMPLIED boolean attribute ***excludeFromSignature***, and two IMPLIED nonNegativeInteger attributes, ***minOccurs*** and ***maxOccurs.***

The ***idref*** attribute has as its value the value of the ***id*** attribute of a previous ***Composite,*** ***Encapsulation,*** or ***SimplePart*** element. The purpose of this sequence of ***Constituents*** is to indicate both the contents and the order of what is packaged within the current ***Composite*** or ***Encapsulation***.

The ***excludeFromSignature*** attribute indicates that this Constituent is not to be included as part of the ebXML message [XMLDSIG] signature. In other words, the signature generated by the *Message Service Handler* should not include a ***ds:Reference*** element to provide a digest for this ***Constituent*** of the *Message*. This attribute is applicable only if the ***Constituent*** is part of the top-level ***Composite*** that corresponds to the entire ebXML *Message*.

The ***minOccurs*** and ***maxOccurs*** attributes serve to specify the value or range of values that the referred to item may occur within ***Composite.*** When unused, it is understood that the item is used exactly once.

The ***Encapsulation*** element is typically employed to indicate the use of MIME security mechanisms, such as [S/MIME] or Open-PGP[RFC2015]. A security body part can encapsulate a MIME part that has been previously characterized. For convenience, all such security structures are under the ***Encapsulation*** element, even when technically speaking the data is not "inside" the body part. (In other words, the so-called clear-signed or detached signature structures possible with MIME multipart/signed are for simplicity found under the ***Encapsulation*** element.)

Another possible use of the ***Encapsulation*** element is to represent the application of a compression algorithm such as gzip [ZLIB] to some part of the payload, prior to its being encrypted and or signed.

The ***Encapsulation*** element has the following attributes:

- a REQUIRED ***mimetype*** attribute,
- a REQUIRED ***id*** attribute,

- an IMPLIED ***mimeparameters*** attribute.

The ***mimetype*** attribute provides the value of the MIME content-type for this *Message* part, such as "application/pkcs7-mime". The ***id*** attribute, type ID, provides a way to refer to this encapsulation if it needs to be mentioned as a constituent of some later element in the sequence. The ***mimeparameters*** attribute provides the values of any significant MIME parameter(s) needed to understand the processing demands of the content-type.

Both the ***Encapsulation*** element and the ***Composite*** element have child elements consisting of a ***Constituent*** element or of a repeatable sequence of ***Constituent*** elements, respectively.

The ***Constituent*** element also has zero or one ***SignatureTransform*** child element and zero or one ***EncryptionTransform*** child element. The ***SignatureTransform*** element is intended for use with XML Digital Signature [XMLDSIG]. When present, it identifies the transforms that must be applied to the source data before a digest is computed. The ***EncryptionTransform*** element is intended for use with XML Encryption [XMLENC]. When present, it identifies the transforms that must be applied to a ***CipherReference*** before decryption can be performed. The ***SignatureTransforms*** element and the ***EncryptionTransforms*** element each contains one or more ***ds:Transform*** [XMLDSIG] elements.

8.7 Signature element

The ***Signature*** element (cardinality zero or one) enables the CPA to be digitally signed using technology that conforms with the XML Digital Signature specification[XMLDSIG]. The ***Signature*** element is the root of a subtree of elements used for signing the *CPP*. The syntax is:

```
<tp:Signature>...</tp:Signature>
```

The ***Signature*** element contains one or more ***ds:Signature*** elements. The content of the ***ds:Signature*** element and any sub-elements are defined by the XML Digital Signature specification. See Section 9.9 for a detailed discussion.

> NOTE: It is necessary to wrap the ***ds:Signature*** elements with a ***Signature*** element in the target namespace to allow for the possibility of having wildcard elements (with namespace="##other") within the ***CollaborationProtocolProfile*** and ***CollaborationProtocolAgreement*** elements. The content model would be ambiguous without the wrapping.

The following additional constraints on ***ds:Signature*** are imposed:

- A *CPP* MUST be considered invalid if any ***ds:Signature*** element fails core validation as defined by the XML Digital Signature specification[XMLDSIG].

- Whenever a *CPP* is signed, each ***ds:Reference*** element within a ***ProcessSpecification*** element MUST pass reference validation and each ***ds:Signature*** element MUST pass core validation.

NOTE: In case a *CPP* is unsigned, software might nonetheless validate the ***ds:Reference*** elements within ***ProcessSpecification*** elements and report any exceptions.

NOTE: Software for creation of *CPPs* and *CPAs* MAY recognize ***ds:Signature*** and automatically insert the element structure necessary to define signing of the *CPP* and *CPA*. Signature generation is outlined in Section 9.9.1.1; details of the cryptographic process are outside the scope of this specification.

NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which validity tests MAY be made.

8.8 Comment element

The ***CollaborationProtocolProfile*** element contains zero or more ***Comment*** elements. The ***Comment*** element is a textual note that can be added to serve any purpose the author desires. The language of the ***Comment*** is identified by a REQUIRED ***xml:lang*** attribute. The ***xml:lang*** attribute MUST comply with the rules for identifying languages specified in [XML]. If multiple ***Comment*** elements are present, each can have a different ***xml:lang*** attribute value. An example of a ***Comment*** element follows:

```
<tp:Comment xml:lang="en-US">This is a CPA between A and B</tp:Comment>
```

When a *CPA* is composed from two *CPPs*, all ***Comment*** elements from both *CPPs* SHALL be included in the *CPA* unless the two *Parties* agree otherwise.

9 CPA Definition

A *Collaboration-Protocol Agreement (CPA)* defines the capabilities that two *Parties* need to agree upon to enable them to engage in electronic *Business* for the purposes of the particular *CPA*. This section defines and discusses the details of the *CPA.* The discussion is illustrated with some XML fragments.

Most of the XML elements in this section are described in detail in Section 8, "CPP Definition". In general, this section does not repeat that information. The discussions in this section are limited to those elements that are not in the *CPP* or for which additional discussion is needed in the *CPA* context. See also Appendix D for the XML Schema, and Appendix B for an example of a *CPA* document.

9.1 CPA Structure

Following is the overall structure of the *CPA:*

```
     <CollaborationProtocolAgreement
        xmlns:tp="http://www.oasis-open.org/committees/ebxml-
cppa/schema/cpp-cpa-2_0.xsd"
         xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         tp:cpaid="YoursAndMyCPA"
         tp:version="2.0a">
         <tp:Status tp:value="proposed"/>
         <tp:Start>1988-04-07T18:39:09</Start>
         <tp:End>1990-04-07T18:40:00</End>
         <!-- ConversationConstraints MAY appear 0 or 1 time -->
         <tp:ConversationConstraints
             tp:invocationLimit="100"
             tp:concurrentConversations="4"/>
         <tp:PartyInfo>
         ...
         </tp:PartyInfo>
         <tp:PartyInfo>
         ...
         </tp:PartyInfo>
         <tp:SimplePart tp:id="...">  <!-- one or more -->
         ...
         </tp:SimplePart>
         <tp:Packaging tp:id="...">   <!-- one or more -->
         ...
         </tp:Packaging>
         <tp:Signature>  <!-- zero or one time -->
          ...
          </tp:Signature>
         <tp:Comment xml:lang="en-GB">any text</Comment> <!--
zero or more -->
    </tp:CollaborationProtocolAgreement>
```

9.2 CollaborationProtocolAgreement element

The ***CollaborationProtocolAgreement*** element is the root element of a *CPA*. It has a REQUIRED ***cpaid*** attribute that supplies a unique identifier for the document. The value of the ***cpaid*** attribute SHALL be assigned by one *Party* and used by both. It is RECOMMENDED that the value of the ***cpaid*** attribute be a URI. The value of the ***cpaid*** attribute SHALL be used as the value of the ***CPAId*** element in the ebXML *Message Header*[ebMS] or of a similar element in a *Message Header* of an alternative messaging service.

NOTE: Each *Party* might associate a local identifier with the ***cpaid*** attribute.

In addition, the ***CollaborationProtocolAgreement*** element has a REQUIRED ***version*** attribute. This attribute indicates the version of the schema to which the *CPA* conforms. The value of the ***version*** attribute SHOULD be a string such as "2_0a", "2_0b", etc.

NOTE: The method of assigning unique ***cpaid*** values is left to the implementation.

The ***CollaborationProtocolAgreement*** element has REQUIRED [XML] Namespace[XMLNS] declarations that are defined in Section 8, "CPP Definition".

The ***CollaborationProtocolAgreement*** element is comprised of the following child elements, most of which are described in greater detail in subsequent sections:

- a REQUIRED ***Status*** element that identifies the state of the process that creates the *CPA,*
- a REQUIRED ***Start*** element that records the date and time that the *CPA* goes into effect,
- a REQUIRED ***End*** element that records the date and time after which the *CPA* MUST be renegotiated by the *Parties,*
- zero or one ***ConversationConstraints*** element that documents certain agreements about conversation processing,
- two REQUIRED ***PartyInfo*** elements, one for each *Party* to the *CPA,*
- one or more ***SimplePart*** elements,
- one or more ***Packaging*** elements,
- zero or one ***Signature*** element that provides for signing of the *CPA* using the XML Digital Signature[XMLDSIG] standard,
- zero or more ***Comment*** elements.

9.3 Status Element

The ***Status*** element records the state of the composition/negotiation process that creates the *CPA*. An example of the ***Status*** element follows:

```
<tp:Status tp:value="proposed"/>
```

The Status element has a REQUIRED ***value*** attribute that records the current state of composition of the *CPA*. This attribute is an enumeration comprised of the following possible values:

- "proposed", meaning that the *CPA* is still being negotiated by the *Parties,*
- "agreed", meaning that the contents of the *CPA* have been agreed to by both *Parties,*
- "signed", meaning that the *CPA* has been "signed" by one or more of the *Parties.* This "signing" takes the form of a digital signature that is described in Section 9.7 below.

NOTE: The ***Status*** element MAY be used by a *CPA* composition and negotiation tool to assist it in the process of building a *CPA*.

NOTE: The value of the ***Status*** element's ***value*** attribute is set to "signed" before the first *Party* signs. Even though excluding ***value*** attribute from a signature might be technically feasible, it is preferable to change the attribute's value to "signed" prior to the first signature, and maintain it as "signed" for any subsequent signatures.

9.4 CPA Lifetime

The lifetime of the *CPA* is given by the ***Start*** and ***End*** elements. The syntax is:

```
<tp:Start>1988-04-07T18:39:09Z</tp:Start>
<tp:End>1990-04-07T18:40:00Z</tp:End>
```

9.4.1 Start element

The ***Start*** element specifies the starting date and time of the *CPA*. The ***Start*** element SHALL be a string value that conforms to the content model of a canonical dateTime type as defined in the XML Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm UTC (Coordinated Universal Time) on May 31, 1999, a ***Start*** element would have the following value:

```
1999-05-31T13:20:00Z
```

The ***Start*** element SHALL be represented as Coordinated Universal Time (UTC).

9.4.2 End element

The ***End*** element specifies the ending date and time of the *CPA*. The ***End*** element SHALL be a string value that conforms to the content model of a canonical dateTime type as defined in the XML Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm UTC (Coordinated Universal Time) on May 31, 1999, an ***End*** element would have the following value:

```
1999-05-31T13:20:00Z
```

The ***End*** element SHALL be represented as Coordinated Universal Time (UTC).

When the end of the *CPA's* lifetime is reached, any *Business Transaction*s that are still in progress SHALL be allowed to complete and no new *Business Transactions* SHALL be started.

When all in-progress *Business Transactions* on each conversation are completed, the *Conversation* SHALL be terminated whether or not it was completed.

When a *CPA* is signed, software for signing the agreements SHALL warn if any signing certificate's validity expires prior to the proposed time for ending the *CPA*. The opportunity to renegotiate a *CPA* ***End*** value or to in some other way align certificate validity periods with CPA validity periods SHALL be made available. (Other ways to align these validity periods would include reissuing the signing certificates for a longer period or obtaining new certificates for this purpose.)

Signing software SHOULD also attempt to align the validity periods of certificates referred to within the CPA that perform security functions so as to not expire before the CPA expires. This alignment can occur in several ways including making use of ***ds:KeyInfo's*** content model ***ds:RetrievalMethod*** so that a new certificate can be installed and still be retrieved in accordance with the information in ***ds:RetrievalMethod.*** If no alignment can be attained, signing software MUST warn the user of the situation that the *CPA* validity exceeds the validity of some of the certificates referred to within the *CPA*.

NOTE: If a *Business* application defines a conversation as consisting of multiple *Business Transactions*, such a conversation MAY be terminated with no error indication when the end of the lifetime is reached. The run-time system could provide an error indication to the application.

NOTE: It might not be feasible to wait for outstanding conversations to terminate before ending the *CPA* since there is no limit on how long a conversation can last.

NOTE: The run-time system SHOULD return an error indication to both *Parties* when a new *Business Transaction* is started under this *CPA* after the date and time specified in the ***End*** element.

9.5 ConversationConstraints Element

The ***ConversationConstraints*** element places limits on the number of conversations under the *CPA*. An example of this element follows:

```
<tp:ConversationConstraints tp:invocationLimit="100"
    tp:concurrentConversations="4"/>
```

The ***ConversationConstraints*** element has the following attributes:

- an IMPLIED ***invocationLimit*** attribute,
- an IMPLIED concurrentConversations attribute.

9.5.1 invocationLimit attribute

The ***invocationLimit*** attribute defines the maximum number of conversations that can be processed under the *CPA*. When this number has been reached, the *CPA* is terminated and

MUST be renegotiated. If no value is specified, there is no upper limit on the number of conversations and the lifetime of the *CPA* is controlled solely by the ***End*** element.

NOTE: The ***invocationLimit*** attribute sets a limit on the number of units of *Business* that can be performed under the *CPA*. It is a *Business* parameter, not a performance parameter. A CPA expires whichever terminating condition (***End*** or ***invocationLimit)*** is first reached.

9.5.2 concurrentConversations attribute

The ***concurrentConversations*** attribute defines the maximum number of conversations that can be in process under this *CPA* at the same time. If no value is specified, processing of concurrent conversations is strictly a local matter.

NOTE: The ***concurrentConversations*** attribute provides a parameter for the *Parties* to use when it is necessary to limit the number of conversations that can be concurrently processed under a particular *CPA*. For example, the back-end process might only support a limited number of concurrent conversations. If a request for a new conversation is received when the maximum number of conversations allowed under this *CPA* is already in process, an implementation MAY reject the new conversation or MAY enqueue the request until an existing conversation ends. If no value is given for ***concurrentConversations***, how to handle a request for a new conversation for which there is no capacity is a local implementation matter.

9.6 PartyInfo Element

The general characteristics of the ***PartyInfo*** element are discussed in Section 8.4.

The *CPA* SHALL have one ***PartyInfo*** element for each *Party* to the *CPA*. The ***PartyInfo*** element specifies the *Parties'* agreed terms for engaging in the *Business Collaborations* defined by the *Process-Specification* documents referenced by the *CPA*. If a *CPP* has more than one ***PartyInfo*** element, the appropriate ***PartyInfo*** element SHALL be selected from each *CPP* when composing a *CPA*.

In the *CPA*, there SHALL be one or more ***PartyId*** elements under each ***PartyInfo*** element. The values of these elements are the same as the values of the ***PartyId*** elements in the ebXML *Message* Service specification[ebMS] or similar messaging service specification. These ***PartyId*** elements SHALL be used within a ***To*** or ***From*** *Header* element of an ebXML *Message*.

9.6.1 ProcessSpecification element

The ***ProcessSpecification*** element identifies the *Business Collaboration* that the two *Parties* have agreed to perform. There can be one or more ***ProcessSpecification*** elements in a *CPA*. Each SHALL be a child element of a separate ***CollaborationRole*** element. See the discussion in Section 8.4.3.

9.7 SimplePart element

The ***CollaborationProtocolAgreement*** element SHALL contain one or more ***SimplePart*** elements. See Section 8.5 for details of the syntax of the ***SimplePart*** element.

9.8 Packaging element

The ***CollaborationProtocolAgreement*** element SHALL contain one or more ***Packaging*** elements. See Section 8.6 for details of the syntax of the ***Packaging*** element.

9.9 Signature element

A *CPA* document can be digitally signed by one or more of the *Parties* as a means of ensuring its integrity as well as a means of expressing the agreement just as a corporate officer's signature would do for a paper document. If signatures are being used to digitally sign an ebXML *CPA* or *CPP* document, then [XMLDSIG] SHALL be used to digitally sign the document.

The ***Signature*** element, if present, is made up of one to three ***ds:Signature*** elements. The *CPA* can be signed by one or both *Parties*. It is RECOMMENDED that both *Parties* sign the *CPA*. For signing by both *Parties*, one *Party* initially signs. The other *Party* then signs over the first *Party's* signature. The resulting *CPA* MAY then be signed by a notary.

The ***ds:Signature*** element is the root of a subtree of elements used for signing the *CPP*.

The content of this element and any sub-elements are defined by the XML Digital Signature specification[XMLDSIG]. The following additional constraints on ***ds:Signature*** are imposed:

- A *CPA* MUST be considered invalid if any ***ds:Signature*** fails core validation as defined by the XML Digital Signature specification.

- Whenever a *CPA* is signed, each ***ds:Reference*** within a ***ProcessSpecification*** MUST pass reference validation and each ***ds:Signature*** MUST pass core validation.

  NOTE: In case a *CPA* is unsigned, software MAY nonetheless validate the ***ds:Reference*** elements within ***ProcessSpecification*** elements and report any exceptions.

Software for creation of *CPPs* and *CPAs* SHALL recognize ***ds:Signature*** and automatically insert the element structure necessary to define signing of the *CPP* and *CPA*. Signature creation itself is a cryptographic process that is outside the scope of this specification.

NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which a *CPA* MAY be validated.

9.9.1 Persistent Digital Signature

If [XMLDSIG] is used to sign an ebXML *CPP* or *CPA,* the process defined in this section of the specification SHALL be used.

9.9.1.1 Signature Generation

Following are the steps to create a digital signature:

1. Create a ***SignedInfo*** element, a child element of ***ds:Signature. SignedInfo*** SHALL have child elements ***SignatureMethod***, ***CanonicalizationMethod***, and ***Reference*** as prescribed by [XMLDSIG].
2. Canonicalize and then calculate the **SignatureValue** over ***SignedInfo*** based on algorithms specified in ***SignedInfo*** as specified in [XMLDSIG].
3. Construct the ***Signature*** element that includes the ***SignedInfo***, ***KeyInfo*** (RECOMMENDED), and ***SignatureValue*** elements as specified in [XMLDSIG].
4. Include the namespace qualified ***Signature*** element in the document just signed, following the last ***PartyInfo*** element.

9.9.1.2 ds:SignedInfo element

The ***ds:SignedInfo*** element SHALL be comprised of zero or one ***ds:CanonicalizationMethod*** element, the ***ds:SignatureMethod*** element, and one or more ***ds:Reference*** elements.

9.9.1.3 ds:CanonicalizationMethod element

The ***ds:CanonicalizationMethod*** element as defined in [XMLDSIG], can occur zero or one time, meaning that the element need not appear in an instance of a ***ds:SignedInfo*** element. The default canonicalization method that is applied to the data to be signed is [XMLC14N] in the absence of a ***ds:CanonicalizationMethod*** element that specifies otherwise. This default SHALL also serve as the default canonicalization method for the ebXML *CPP* and *CPA* documents.

9.9.1.4 ds:SignatureMethod element

The ***ds:SignatureMethod*** element SHALL be present and SHALL have an ***Algorithm*** attribute. The RECOMMENDED value for the ***Algorithm*** attribute is:

```
"http://www.w3.org/2000/09/xmldsig#sha1"
```

This RECOMMENDED value SHALL be supported by all compliant ebXML *CPP* or *CPA* software implementations.

9.9.1.5 ds:Reference element

The ***ds:Reference*** element for the *CPP* or *CPA* document SHALL have a REQUIRED URI attribute value of "" to provide for the signature to be applied to the document that contains the ***ds:Signature*** element (the *CPA* or *CPP* document). The ***ds:Reference*** element for the *CPP* or *CPA* document can include an IMPLIED ***type*** attribute that has a value of:

```
"http://www.w3.org/2000/09/xmldsig#Object"
```

in accordance with [XMLDSIG]. This attribute is purely informative. It MAY be omitted. Implementations of software designed to author or process an ebXML *CPA* or *CPP* document SHALL be prepared to handle either case. The ***ds:Reference*** element can include the ***id*** attribute, type ID, by which this ***ds:Reference*** element is referenced from a ***ds:Signature*** element.

9.9.1.6 ds:Transform element

The ***ds:Reference*** element for the *CPA* or *CPP* document SHALL include a descendant ***ds:Transform*** element that excludes the containing ***ds:Signature*** element and all its descendants. This exclusion is achieved by means of specifying the ***ds:Algorithm*** attribute of the ***Transform*** element as

```
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"
```

For example:

```
<ds:Reference ds:URI="">
    <ds:Transforms>
        <ds:Transform
ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
    </ds:Transforms>
     <ds:DigestMethod
 ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <ds:DigestValue>...</ds:DigestValue>
</ds:Reference>
```

9.9.1.7 ds:Algorithm attribute

The ***ds:Transform*** element SHALL include a ds:***Algorithm*** attribute that has a value of:

```
http://www.w3.org/2000/09/xmldsig#enveloped-signature
```

NOTE: When digitally signing a *CPA*, it is RECOMMENDED that each *Party* sign the document in accordance with the process described above.

When the two *Parties* sign the *CPA*, the first *Party* that signs the *CPA* SHALL sign only the *CPA* contents, excluding their own signature. The second *Party* SHALL sign over the contents of the *CPA* as well as the ***ds:Signature*** element that contains the first *Party's* signature. If necessary, a notary can then sign over both signatures.

9.10 Comment element

The ***CollaborationProtocolAgreement*** element contains zero or more ***Comment*** elements. See Section 8.8 for details of the syntax of the ***Comment*** element.

9.11 Composing a CPA from Two CPPs

This section discusses normative issues in composing a *CPA* from two *CPPs*. See also Appendix E, "CPA Composition (Non-Normative)".

9.11.1 ID Attribute Duplication

In composing a *CPA* from two *CPPs*, there is a hazard that ID attributes from the two *CPPs* might have duplicate values. When a *CPA* is composed from two *CPPs*, duplicate ID attribute values SHALL be tested for. If a duplicate ID attribute value is present, one of the duplicates SHALL be given a new value and the corresponding IDREF attribute values from the corresponding *CPP* SHALL be corrected.

NOTE: A party can seek to prevent ID/IDREF reassignment in the *CPA* by choosing ID and IDREF values which are likely to be unique among its trading partners. For example, the following ***Certificate*** element found in a *CPP* has a ***certId*** attribute that is generic enough that it might clash with a ***certId*** attribute found in a collaborating party's *CPP*:

```
<tp:Certificate
tp:certId="EncryptionCert"><ds:KeyInfo/></tp:Certificate>
```

To prevent reassignment of this ID (and its associated IDREFs) in a *CPA*, a better choice of ***certId*** in Company A's *CPP* would be:

```
<tp:Certificate
tp:certId="CompanyA_EncryptionCert"><ds:KeyInfo/></tp:Certificate>
```

9.12 Modifying Parameters of the Process-Specification Document Based on Information in the CPA

A *Process-Specification* document contains a number of parameters, expressed as XML attributes. An example is the security attributes that are counterparts of the attributes of the *CPA* ***BusinessTransactionCharacteristics*** element. The values of these attributes can be considered to be default values or recommendations. When a *CPA* is created, the *Parties* might decide to accept the recommendations in the *Process-Specification* or they MAY agree on values of these parameters that better reflect their needs.

When a *CPA* is used to configure a run-time system, choices specified in the *CPA* MUST always assume precedence over choices specified in the referenced *Process-Specification* document. In particular, all choices expressed in a *CPA's* ***BusinessTransactionCharacteristics*** and ***Packaging*** elements MUST be implemented as agreed to by the *Parties*. These choices SHALL override the default values expressed in the *Process-Specification* document. The process of installing the information from the *CPA* and *Process-Specification* document MUST verify that all of the resulting choices are mutually consistent and MUST signal an error if they are not.

NOTE: There are several ways of overriding the information in the *Process-Specification* document by information from the *CPA*. For example:

- The *CPA* composition tool can create a separate copy of the Process-Specification

document. The tool can then directly modify the *Process-Specification* document with information from the *CPA*. An advantage of this method is that the override process is performed entirely by the *CPA* composition tool.

- A *CPA* installation tool can dynamically override parameters in the *Process-Specification* document using information from the corresponding parameters in the *CPA* at the time the *CPA* and *Process-Specification* document are installed in the *Parties'* systems. This eliminates the need to create a separate copy of the *Process-Specification* document.
- Other possible methods might be based on XSLT transformations of the parameter information in the *CPA* and/or the *Process-Specification* document.

10 References

Some references listed below specify functions for which specific XML definitions are provided in the *CPP* and *CPA*. Other specifications are referred to in this specification in the sense that they are represented by keywords for which the *Parties* to the *CPA* MAY obtain plug-ins or write custom support software but do not require specific XML element sets in the *CPP* and *CPA*.

In a few cases, the only available specification for a function is a proprietary specification. These are indicated by notes within the citations below.

[ccOVER] ebXML Core Components Overview, http://www.ebxml.org/specs/ccOVER.pdf.

[DIGENV] Digital Envelope, RSA Laboratories, http://www.rsasecurity.com/rsalabs/faq/2-2-4.html.
NOTE: At this time, the only available specification for digital envelope appears to be the RSA Laboratories specification.

[ebBPSS] ebXML Business Process Specification Schema, http://www.ebxml.org/specs/ebBPSS.pdf.

[ebMS] ebXML Message Service Specification, http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.

[ebRS] ebXML Registry Services Specification, http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf.

[HTTP] Hypertext Transfer Protocol, Internet Engineering Task Force RFC 2616, http://www.rfc-editor.org/rfc/rfc2616.txt.

[IPSEC] IP Security Document Roadmap, Internet Engineering Task Force RFC 2411, http://www.ietf.org/rfc/rfc2411.txt.

[ISO6523] Structure for the Identification of Organizations and Organization Parts, International Standards Organization ISO-6523.

[MIME] MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet *Message* Bodies. Internet Engineering Task Force RFC 1521, http://www.ietf.org/rfc/rfc1521.txt.

[RFC959] File Transfer Protocol (FTP), Internet Engineering Task Force RFC 959, http://www.ietf.org/rfc/rfc959.txt.

[RFC1123] Requirements for Internet Hosts -- Application and Support, Internet Engineering Task Force RFC 1123, http://www.ietf.org/rfc/rfc1123.txt.

[RFC1579] Firewall-Friendly FTP, Internet Engineering Task Force RFC 1579, http://www.ietf.org/rfc/rfc1579.txt.

[RFC2015] MIME Security with Pretty Good Privacy, Internet Engineering Task Force, RFC 2015, http://www.ietf.org/rfc/rfc2015.txt.

[RFC2119] Key Words for use in RFCs to Indicate Requirement Levels, Internet Engineering Task Force RFC 2119, http://www.ietf.org/rfc/rfc2119.txt.

[RFC2246] The TLS Protocol, Internet Engineering Task Force RFC 2246, http://www.ietf.org/rfc/rfc2246.txt.

[RFC2251] Lightweight Directory Access Protocol (v3), Internet Engineering Task Force RFC 2251, http://www.ietf.org/rfc/rfc2251.txt.

[RFC2396] Uniform Resource Identifiers (URI): Generic Syntax, Internet Engineering Task Force RFC 2396, http://www.ietf.org/rfc/rfc2396.txt.

[RFC2617] HTTP Authentication: Basic and Digest Authentication, , Internet Engineering Task Force RFC 2617, http://www.ietf.org/rfc/rfc2617.txt.

[RFC2822] Internet Message Format, Internet Engineering Task Force RFC 2822, http://www.ietf.org/rfc/rfc2822.txt.

[S/MIME] S/MIME Version 3 Message Specification, Internet Engineering Task Force RFC 2633, http://www.ietf.org/rfc/rfc2633.txt.

[SAML] Security Assertion Markup Language, http://www.oasis-open.org/committees/security/ - documents.

[SMTP] Simple Mail Transfer Protocol, Internet Engineering Task Force RFC 2821, http://www.faqs.org/rfcs/rfc2821.html.

[SSL] Secure Sockets Layer, Netscape Communications Corp., http://www.netscape.com/eng/ssl3/ NOTE: At this time, it appears that the Netscape specification is the only available specification of SSL.

[X12] ANSI X12 Standard for Electronic Data Interchange, X12 Standard Release 4050, December 2001.

[XAML] Transaction Authority Markup Language, http://xaml.org/.

[XLINK] XML Linking Language, http://www.w3.org/TR/xlink/.

[XML] Extensible Markup Language (XML), World Wide Web Consortium,

http://www.w3.org/XML.

[XMLC14N] Canonical XML, Ver. 1.0, Worldwide Web Consortium, http://www.w3.org/TR/2001/REC-xml-c14n-20010315.

[XMLDSIG] XML Signature Syntax and Processing, Worldwide Web Consortium, http://www.w3.org/TR/xmldsig-core/.

[XMLENC] XML Encryption Syntax and Processing, Worldwide Web Consortium, http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/.

[XMLNS] Namespaces in XML, Worldwide Web Consortium, http://www.w3.org/TR/REC-xml-names/.

[XMLSCHEMA-1] XML Schema Part 1: Structures, Worldwide Web Consortium, http://www.w3.org/TR/xmlschema-1/.

[XMLSCHEMA-2] XML Schema Part 2: Datatypes, Worldwide Web Consortium, http://www.w3.org/TR/xmlschema-2/.

[XPOINTER] XML Pointer Language, Worldwide Web Consortium, http://www.w3.org/TR/xptr/.

[ZLIB] Zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library, http://www.gzip.org/zlib/.

11 Conformance

In order to conform to this specification, an implementation:

a) SHALL support all the functional and interface requirements defined in this specification,
b) SHALL NOT specify any requirements that would contradict or cause non-conformance to this specification.

A conforming implementation SHALL satisfy the conformance requirements of the applicable parts of this specification.

An implementation of a tool or service that creates or maintains ebXML *CPP* or *CPA* instance documents SHALL be determined to be conformant by validation of the *CPP* or *CPA* instance documents, created or modified by said tool or service, against the XML Schema[XMLSCHEMA-1] definition of the *CPP* or *CPA* in Appendix D and available from

http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd

by using two or more validating XML Schema parsers that conform to the W3C XML Schema specifications[XMLSCHEMA-1, XMLSCHEMA-2].

The objective of conformance testing is to determine whether an implementation being tested conforms to the requirements stated in this specification. Conformance testing enables vendors to implement compatible and interoperable systems. Implementations and applications SHALL be tested using available test suites to verify their conformance to this specification.

Publicly available test suites from vendor neutral organizations such as OASIS and the U.S.A. National Institute of Science and Technology (NIST) SHOULD be used to verify the conformance of implementations, applications, and components claiming conformance to this specification. Open-source reference implementations might be available to allow vendors to test their products for interface compatibility, conformance, and interoperability.

3413

3414
3415
3416
3417

12 Contact Information

Arvola Chan (Author)
TIBCO Software
3303 Hillview Avenue
Palo Alto, CA 94304
USA
Phone: 650-846-5046
email: mailto:arvola@tibco.com

Dale W. Moberg (Author)
Cyclone Commerce
8388 E. Hartford Drive
Scottsdale, AZ 85255
USA
Phone: 480-627-2648
email: mailto:dmoberg@cyclonecommerce.com

Himagiri Mukkamala (Author)
Sybase Inc.
5000 Hacienda Dr
Dublin, CA, 84568
USA
Phone: 925-236-5477
email: mailto:himagiri@sybase.com

Peter M. Ogden (Author)
Cyclone Commerce, Inc.
8388 East Hartford Drive
Scottsdale, AZ 85255
USA
Phone: 480-627-1800
email: mailto:pogden@cyclonecommerce.com

Martin W. Sachs (Author)
IBM T. J. Watson Research Center
P.O.B. 704
Yorktown Hts, NY 10598
USA
Phone: 914-784-7287
email: mailto:mwsachs@us.ibm.com

Tony Weida (Coordinating Editor)

3461 535 West 110th St., #4J
3462 New York, NY 10025
3463 USA
3464 Phone: 212-678-5265
3465 email: mailto:rweida@hotmail.com
3466
3467 Jean Zheng
3468 Vitria
3469 945 Stewart Drive
3470 Sunnyvale, CA 94086
3471 USA
3472 Phone: 408-212-2468
3473 email: mailto:jzheng@vitria.com

(Blank page)

Appendix A Example of CPP Document (Non-Normative)

This example includes two CPPs that are used to form the CPA in Appendix B. They are available as ASCII files at

http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-example-companyA-2_0b.xml

http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-example-companyB-2_0b.xml

cpp-example-companyA-2_0b.xml:

```
<?xml version="1.0"?>
<!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
<tp:CollaborationProtocolProfile
  xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
                      cpp-cpa-2_0.xsd"
  tp:cppid="uri:companyA-cpp" tp:version="2_0b">
  <!-- Party info for CompanyA-->
  <tp:PartyInfo
    tp:partyName="CompanyA"
    tp:defaultMshChannelId="asyncChannelA1"
    tp:defaultMshPackageId="CompanyA_MshSignalPackage">
    <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:CollaborationRole>
      <tp:ProcessSpecification
        tp:version="2.0"
        tp:name="PIP3A4RequestPurchaseOrder"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml"
        tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
      <tp:Role
        tp:name="Buyer"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
      <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
      <tp:ServiceBinding>
        <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID1"
            tp:action="Purchase Order Request Action"
            tp:packageId="CompanyA_RequestPackage">
            <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"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanSend>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID2"
            tp:action="ReceiptAcknowledgement"
            tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
```

```
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
                          tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanSend>
        <!-- The next binding uses a synchronous delivery channel -->
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID6"
            tp:action="Purchase Order Request Action"
            tp:packageId="CompanyA_RequestPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
                          tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT5M"
              tp:timeToPerform="PT5M"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Request Action"/>
            <tp:ChannelId>syncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:CanReceive>
            <tp:ThisPartyActionBinding
              tp:id="companyA_ABID7"
              tp:action="Purchase Order Confirmation Action"
              tp:packageId="CompanyA_SyncReplyPackage">
              <tp:BusinessTransactionCharacteristics
                tp:isNonRepudiationRequired="true"
                tp:isNonRepudiationReceiptRequired="true"
                tp:isConfidential="transient"
                tp:isAuthenticated="persistent"
                tp:isTamperProof="persistent"
                tp:isAuthorizationRequired="true"
                tp:timeToAcknowledgeReceipt="PT5M"/>
              <tp:ActionContext
                tp:binaryCollaboration="Request Purchase Order"
                tp:businessTransactionActivity="Request Purchase Order"
                tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
              <tp:ChannelId>syncChannelA1</tp:ChannelId>
            </tp:ThisPartyActionBinding>
          </tp:CanReceive>
          <tp:CanReceive>
            <tp:ThisPartyActionBinding
              tp:id="companyA_ABID8"
              tp:action="Exception"
              tp:packageId="CompanyA_ExceptionPackage">
              <tp:BusinessTransactionCharacteristics
                tp:isNonRepudiationRequired="true"
                tp:isNonRepudiationReceiptRequired="true"
                tp:isConfidential="transient"
                tp:isAuthenticated="persistent"
                tp:isTamperProof="persistent"
                tp:isAuthorizationRequired="true"/>
              <tp:ChannelId>syncChannelA1</tp:ChannelId>
            </tp:ThisPartyActionBinding>
          </tp:CanReceive>
        </tp:CanSend>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID3"
            tp:action="Purchase Order Confirmation Action"
            tp:packageId="CompanyA_ResponsePackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
```

```
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT2H"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanReceive>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID4"
            tp:action="ReceiptAcknowledgment"
            tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent" tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanReceive>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID5"
            tp:action="Exception"
            tp:packageId="CompanyA_ExceptionPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanReceive>
      </tp:ServiceBinding>
    </tp:CollaborationRole>
    <!-- Certificates used by the "Buyer" company -->
    <tp:Certificate tp:certId="CompanyA_AppCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_SigningCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_EncryptionCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_ServerCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_ClientCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA1">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA1_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA2">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA3">
      <ds:KeyInfo>
```

```
        <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA4">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA5">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <!-- An asynchronous delivery channel -->
    <tp:DeliveryChannel
      tp:channelId="asyncChannelA1"
      tp:transportId="transportA2"
      tp:docExchangeId="docExchangeA1">
      <tp:MessagingCharacteristics
        tp:syncReplyMode="none"
        tp:ackRequested="always"
        tp:ackSignatureRequested="always"
        tp:duplicateElimination="always"/>
    </tp:DeliveryChannel>
    <!-- A synchronous delivery channel -->
    <tp:DeliveryChannel
      tp:channelId="syncChannelA1"
      tp:transportId="transportA1"
      tp:docExchangeId="docExchangeA1">
      <tp:MessagingCharacteristics
        tp:syncReplyMode="signalsAndResponse"
        tp:ackRequested="always"
        tp:ackSignatureRequested="always"
        tp:duplicateElimination="always"/>
    </tp:DeliveryChannel>
    <tp:Transport tp:transportId="transportA1">
      <tp:TransportSender>
        <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
        <tp:AccessAuthentication>basic</tp:AccessAuthentication>
        <tp:AccessAuthentication>digest</tp:AccessAuthentication>
        <tp:TransportClientSecurity>
          <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
          <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
          <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
        </tp:TransportClientSecurity>
      </tp:TransportSender>
      <tp:TransportReceiver>
        <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:TransportServerSecurity>
      </tp:TransportReceiver>
    </tp:Transport>
    <tp:Transport tp:transportId="transportA2">
      <tp:TransportSender>
        <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
        <tp:AccessAuthentication>basic</tp:AccessAuthentication>
        <tp:AccessAuthentication>digest</tp:AccessAuthentication>
```

```
        <tp:TransportClientSecurity>
          <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
          <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
          <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
        </tp:TransportClientSecurity>
      </tp:TransportSender>
      <tp:TransportReceiver>
        <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:TransportServerSecurity>
      </tp:TransportReceiver>
    </tp:Transport>
    <tp:DocExchange tp:docExchangeId="docExchangeA1">
      <tp:ebXMLSenderBinding tp:version="2.0">
        <tp:ReliableMessaging>
          <tp:Retries>3</tp:Retries>
          <tp:RetryInterval>PT2H</tp:RetryInterval>
          <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
        </tp:ReliableMessaging>
        <tp:PersistDuration>P1D</tp:PersistDuration>
        <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:ebXMLSenderBinding>
      <tp:ebXMLReceiverBinding tp:version="2.0">
        <tp:ReliableMessaging>
          <tp:Retries>3</tp:Retries>
          <tp:RetryInterval>PT2H</tp:RetryInterval>
          <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
        </tp:ReliableMessaging>
        <tp:PersistDuration>P1D</tp:PersistDuration>
        <tp:ReceiverNonRepudiation>

<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:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
        </tp:ReceiverNonRepudiation>
        <tp:ReceiverDigitalEnvelope>
          <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
          <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
          <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
        </tp:ReceiverDigitalEnvelope>
      </tp:ebXMLReceiverBinding>
    </tp:DocExchange>
  </tp:PartyInfo>
  <!-- SimplePart corresponding to the SOAP Envelope -->
  <tp:SimplePart
    tp:id="CompanyA_MsgHdr"
    tp:mimetype="text/xml">
    <tp:NamespaceSupported
      tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
      tp:version="2.0">
      http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
  <tp:SimplePart
    tp:id="CompanyA_ReceiptAcknowledgment"
```

```
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
        tp:version="2.0">
        http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <!-- SimplePart corresponding to an Exception business signal -->
    <tp:SimplePart
      tp:id="CompanyA_Exception"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
        tp:version="2.0">
        http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <!-- SimplePart corresponding to a request action -->
    <tp:SimplePart
      tp:id="CompanyA_Request"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
        tp:version="2.0">
        http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <!-- SimplePart corresponding to a response action -->
    <tp:SimplePart
      tp:id="CompanyA_Response"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
        tp:version="2.0">
      http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
    </tp:NamespaceSupported>
    </tp:SimplePart>
    <!-- An ebXML message with a SOAP Envelope only -->
    <tp:Packaging tp:id="CompanyA_MshSignalPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_MshSignal"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
    <tp:Packaging tp:id="CompanyA_RequestPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_RequestMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_Request"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
    <tp:Packaging tp:id="CompanyA_ResponsePackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_ResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_Response"/>
```

```
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
         or an ebXML message with an Exception signal -->
    <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_SignalAndResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
          <tp:Constituent tp:idref="CompanyA_Response"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a ReceiptAcknowledgment payload -->
    <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_ReceiptAcknowledgmentMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
    <tp:Packaging tp:id="CompanyA_ExceptionPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_ExceptionMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_Exception"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Comment xml:lang="en-US">Buyer's Collaboration Protocol Profile</tp:Comment>
</tp:CollaborationProtocolProfile>
```

cpp-example-companyB-2_0b.xml:

```
<?xml version="1.0"?>
<!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
<tp:CollaborationProtocolProfile
  xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
                      cpp-cpa-2_0.xsd"
  tp:cppid="uri:companyB-cpp"
  tp:version="2_0b">
  <!-- Party info for CompanyB-->
  <tp:PartyInfo
    tp:partyName="CompanyB"
    tp:defaultMshChannelId="asyncChannelB1"
    tp:defaultMshPackageId="CompanyB_MshSignalPackage">
    <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
    <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html"/>
    <tp:CollaborationRole>
      <tp:ProcessSpecification
        tp:version="2.0"
```

```
        tp:name="PIP3A4RequestPurchaseOrder"
        xlink:type="simple" xlink:href="http://www.rosettanet.org/processes/3A4.xml"
        tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
      <tp:Role
        tp:name="Seller"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
      <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert"/>
      <tp:ServiceBinding>
        <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID1"
            tp:action="Purchase Order Confirmation Action"
            tp:packageId="CompanyB_ResponsePackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT2H"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanSend>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID2"
            tp:action="ReceiptAcknowledgement"
            tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanSend>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID3"
            tp:action="Exception"
            tp:packageId="CompanyB_ExceptionPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanSend>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID4"
            tp:action="Purchase Order Request Action"
            tp:packageId="CompanyB_RequestPackage">
            <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"/>
```

```
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanReceive>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID5" tp:action="ReceiptAcknowledgment"
            tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
        </tp:CanReceive>
        <!-- The next binding uses a synchronous delivery channel -->
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID6"
            tp:action="Purchase Order Request Action"
            tp:packageId="CompanyB_SyncReplyPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT5M"
              tp:timeToPerform="PT5M"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Request Action"/>
            <tp:ChannelId>syncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:CanSend>
            <tp:ThisPartyActionBinding
              tp:id="companyB_ABID7"
              tp:action="Purchase Order Confirmation Action"
              tp:packageId="CompanyB_ResponsePackage">
              <tp:BusinessTransactionCharacteristics
                tp:isNonRepudiationRequired="true"
                tp:isNonRepudiationReceiptRequired="true"
                tp:isConfidential="transient"
                tp:isAuthenticated="persistent"
                tp:isTamperProof="persistent"
                tp:isAuthorizationRequired="true"
                tp:timeToAcknowledgeReceipt="PT5M"/>
              <tp:ActionContext
                tp:binaryCollaboration="Request Purchase Order"
                tp:businessTransactionActivity="Request Purchase Order"
                tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
              <tp:ChannelId>syncChannelB1</tp:ChannelId>
            </tp:ThisPartyActionBinding>
          </tp:CanSend>
          <tp:CanSend>
            <tp:ThisPartyActionBinding
              tp:id="companyB_ABID8"
              tp:action="Exception"
              tp:packageId="CompanyB_ExceptionPackage">
              <tp:BusinessTransactionCharacteristics
                tp:isNonRepudiationRequired="true"
                tp:isNonRepudiationReceiptRequired="true"
                tp:isConfidential="transient"
                tp:isAuthenticated="persistent"
                tp:isTamperProof="persistent"
                tp:isAuthorizationRequired="true"/>
              <tp:ChannelId>syncChannelB1</tp:ChannelId>
            </tp:ThisPartyActionBinding>
          </tp:CanSend>
        </tp:CanReceive>
      </tp:ServiceBinding>
    </tp:CollaborationRole>
    <!-- Certificates used by the "Seller" company -->
    <tp:Certificate tp:certId="CompanyB_AppCert">
```

```
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_SigningCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_EncryptionCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_ServerCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_ClientCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB4">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB5">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB6">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB7">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB8">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <!-- An asynchronous delivery channel -->
    <tp:DeliveryChannel
      tp:channelId="asyncChannelB1"
      tp:transportId="transportB1"
      tp:docExchangeId="docExchangeB1">
      <tp:MessagingCharacteristics
        tp:syncReplyMode="none"
        tp:ackRequested="always"
        tp:ackSignatureRequested="always"
        tp:duplicateElimination="always"/>
    </tp:DeliveryChannel>
    <!-- A synchronous delivery channel -->
    <tp:DeliveryChannel
      tp:channelId="syncChannelB1"
      tp:transportId="transportB2"
      tp:docExchangeId="docExchangeB1">
```

```
        <tp:MessagingCharacteristics
          tp:syncReplyMode="signalsAndResponse"
          tp:ackRequested="always"
          tp:ackSignatureRequested="always"
          tp:duplicateElimination="always"/>
      </tp:DeliveryChannel>
      <tp:Transport tp:transportId="transportB1">
        <tp:TransportSender>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:TransportClientSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
            <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportClientSecurity>
        </tp:TransportSender>
        <tp:TransportReceiver>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:Endpoint
            tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
            tp:type="allPurpose"/>
          <tp:TransportServerSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
            <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportServerSecurity>
        </tp:TransportReceiver>
      </tp:Transport>
      <tp:Transport tp:transportId="transportB2">
        <tp:TransportSender>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:TransportClientSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
            <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportClientSecurity>
        </tp:TransportSender>
        <tp:TransportReceiver>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:Endpoint
            tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
            tp:type="allPurpose"/>
          <tp:TransportServerSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
            <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportServerSecurity>
        </tp:TransportReceiver>
      </tp:Transport>
      <tp:DocExchange tp:docExchangeId="docExchangeB1">
        <tp:ebXMLSenderBinding tp:version="2.0">
          <tp:ReliableMessaging>
            <tp:Retries>3</tp:Retries>
            <tp:RetryInterval>PT2H</tp:RetryInterval>
            <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
          </tp:ReliableMessaging>
          <tp:PersistDuration>P1D</tp:PersistDuration>
          <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="CompanyB_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="CompanyB_MessageSecurity"/>
          </tp:SenderDigitalEnvelope>
        </tp:ebXMLSenderBinding>
        <tp:ebXMLReceiverBinding tp:version="2.0">
          <tp:ReliableMessaging>
            <tp:Retries>3</tp:Retries>
            <tp:RetryInterval>PT2H</tp:RetryInterval>
```

```
          <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
        </tp:ReliableMessaging>
        <tp:PersistDuration>P1D</tp:PersistDuration>
        <tp:ReceiverNonRepudiation>

<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:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
        </tp:ReceiverNonRepudiation>
        <tp:ReceiverDigitalEnvelope>
          <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
          <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
          <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert"/>
        </tp:ReceiverDigitalEnvelope>
      </tp:ebXMLReceiverBinding>
    </tp:DocExchange>
  </tp:PartyInfo>
  <!-- SimplePart corresponding to the SOAP Envelope -->
  <tp:SimplePart
    tp:id="CompanyB_MsgHdr"
    tp:mimetype="text/xml">
    <tp:NamespaceSupported
      tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
      tp:version="2.0">
      http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
  <tp:SimplePart
    tp:id="CompanyB_ReceiptAcknowledgment"
    tp:mimetype="application/xml">
    <tp:NamespaceSupported
      tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
      tp:version="2.0">
      http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- SimplePart corresponding to an Exception business signal -->
  <tp:SimplePart
    tp:id="CompanyB_Exception"
    tp:mimetype="application/xml">
    <tp:NamespaceSupported
      tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
      tp:version="2.0">
      http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- SimplePart corresponding to a request action -->
  <tp:SimplePart
    tp:id="CompanyB_Request"
    tp:mimetype="application/xml">
    <tp:NamespaceSupported
      tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
      tp:version="2.0">
      http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- SimplePart corresponding to a response action -->
  <tp:SimplePart
    tp:id="CompanyB_Response"
    tp:mimetype="application/xml">
    <tp:NamespaceSupported
      tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd.xsd"
      tp:version="2.0">
      http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- An ebXML message with a SOAP Envelope only -->
  <tp:Packaging tp:id="CompanyB_MshSignalPackage">
    <tp:ProcessingCapabilities
      tp:parse="true"
      tp:generate="true"/>
    <tp:CompositeList>
      <tp:Composite
        tp:id="CompanyB_MshSignal"
        tp:mimetype="multipart/related"
```

```
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
    <tp:Packaging tp:id="CompanyB_RequestPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="RequestMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_Request"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
    <tp:Packaging tp:id="CompanyB_ResponsePackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_ResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_Response"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
    <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_ReceiptAcknowledgmentMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
    <tp:Packaging tp:id="CompanyB_ExceptionPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_ExceptionMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_Exception"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
         or an ebXML message with an Exception signal -->
    <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_SignalAndResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
```

```
          <tp:Constituent tp:idref="CompanyB_Response"/>
        </tp:Composite>
      </tp:CompositeList>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_SyncExceptionMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_Exception"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Comment xml:lang="en-US">Seller's Collaboration Protocol Profile</tp:Comment>
</tp:CollaborationProtocolProfile>
```

Appendix B Example of CPA Document (Non-Normative)

The example in this appendix is to be parsed with an XML Schema parser. The schema is available as an ASCII file at

http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0b.xsd

The example that can be parsed with the XSD is available at

http://www.oasis-open.org/committees/ebxml-cppa/schema/cpa-example-2_0b.xml

```
<?xml version="1.0"?>
<!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
<tp:CollaborationProtocolAgreement
  xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
                      cpp-cpa-2_0.xsd"
  tp:cpaid="uri:companyA-and-companyB-cpa" tp:version="2_0b">
  <tp:Status tp:value="proposed"/>
  <tp:Start>2001-05-20T07:21:00Z</tp:Start>
  <tp:End>2002-05-20T07:21:00Z</tp:End>
  <tp:ConversationConstraints tp:invocationLimit="100" tp:concurrentConversations="10"/>
  <!-- Party info for CompanyA -->
  <tp:PartyInfo
    tp:partyName="CompanyA"
    tp:defaultMshChannelId="asyncChannelA1"
    tp:defaultMshPackageId="CompanyA_MshSignalPackage">
    <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:CollaborationRole>
      <tp:ProcessSpecification
        tp:version="2.0"
        tp:name="PIP3A4RequestPurchaseOrder"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml"
        tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
      <tp:Role
        tp:name="Buyer"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
      <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
      <tp:ServiceBinding>
        <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID1"
            tp:action="Purchase Order Request Action"
            tp:packageId="CompanyA_RequestPackage">
            <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"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyB_ABID4</tp:OtherPartyActionBinding>
        </tp:CanSend>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID2"
```

```
            tp:action="ReceiptAcknowledgement"
            tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyB_ABID5</tp:OtherPartyActionBinding>
        </tp:CanSend>
        <!-- The next binding uses a synchronous delivery channel -->
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID6"
            tp:action="Purchase Order Request Action"
            tp:packageId="CompanyA_RequestPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT5M"
              tp:timeToPerform="PT5M"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Request Action"/>
            <tp:ChannelId>syncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyB_ABID6</tp:OtherPartyActionBinding>
          <tp:CanReceive>
            <tp:ThisPartyActionBinding
              tp:id="companyA_ABID7"
              tp:action="Purchase Order Confirmation Action"
              tp:packageId="CompanyA_SyncReplyPackage">
              <tp:BusinessTransactionCharacteristics
                tp:isNonRepudiationRequired="true"
                tp:isNonRepudiationReceiptRequired="true"
                tp:isConfidential="transient"
                tp:isAuthenticated="persistent"
                tp:isTamperProof="persistent"
                tp:isAuthorizationRequired="true"
                tp:timeToAcknowledgeReceipt="PT5M"/>
              <tp:ActionContext
                tp:binaryCollaboration="Request Purchase Order"
                tp:businessTransactionActivity="Request Purchase Order"
                tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
              <tp:ChannelId>syncChannelA1</tp:ChannelId>
            </tp:ThisPartyActionBinding>
            <tp:OtherPartyActionBinding>companyB_ABID7</tp:OtherPartyActionBinding>
          </tp:CanReceive>
          <tp:CanReceive>
            <tp:ThisPartyActionBinding
              tp:id="companyA_ABID8"
              tp:action="Exception"
              tp:packageId="CompanyA_ExceptionPackage">
              <tp:BusinessTransactionCharacteristics
                tp:isNonRepudiationRequired="true"
                tp:isNonRepudiationReceiptRequired="true"
                tp:isConfidential="transient"
                tp:isAuthenticated="persistent"
                tp:isTamperProof="persistent"
                tp:isAuthorizationRequired="true"/>
              <tp:ChannelId>syncChannelA1</tp:ChannelId>
            </tp:ThisPartyActionBinding>
            <tp:OtherPartyActionBinding>companyB_ABID8</tp:OtherPartyActionBinding>
          </tp:CanReceive>
        </tp:CanSend>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID3"
            tp:action="Purchase Order Confirmation Action"
            tp:packageId="CompanyA_ResponsePackage">
```

```
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT2H"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyB_ABID1</tp:OtherPartyActionBinding>
        </tp:CanReceive>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID4"
            tp:action="ReceiptAcknowledgment"
            tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyB_ABID2</tp:OtherPartyActionBinding>
        </tp:CanReceive>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyA_ABID5"
            tp:action="Exception"
            tp:packageId="CompanyA_ExceptionPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyB_ABID3</tp:OtherPartyActionBinding>
        </tp:CanReceive>
      </tp:ServiceBinding>
    </tp:CollaborationRole>
    <!-- Certificates used by the "Buyer" company -->
    <tp:Certificate tp:certId="CompanyA_AppCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_SigningCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_EncryptionCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_ServerCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyA_ClientCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA1">
      <ds:KeyInfo>
```

```
        <ds:KeyName>TrustedRootCertA1_Key </ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA2">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA3">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA4">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertA5">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <tp:DeliveryChannel
      tp:channelId="asyncChannelA1"
      tp:transportId="transportA1"
      tp:docExchangeId="docExchangeA1">
      <tp:MessagingCharacteristics
        tp:syncReplyMode="none"
        tp:ackRequested="always"
        tp:ackSignatureRequested="always"
        tp:duplicateElimination="always"/>
    </tp:DeliveryChannel>
    <tp:DeliveryChannel
      tp:channelId="syncChannelA1"
      tp:transportId="transportA2"
      tp:docExchangeId="docExchangeA1">
      <tp:MessagingCharacteristics
        tp:syncReplyMode="signalsAndResponse"
        tp:ackRequested="always"
        tp:ackSignatureRequested="always"
        tp:duplicateElimination="always"/>
    </tp:DeliveryChannel>
    <tp:Transport tp:transportId="transportA1">
      <tp:TransportSender>
        <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
        <tp:AccessAuthentication>basic</tp:AccessAuthentication>
        <tp:TransportClientSecurity>
          <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
          <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
          <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
        </tp:TransportClientSecurity>
      </tp:TransportSender>
      <tp:TransportReceiver>
        <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
        <tp:AccessAuthentication>basic</tp:AccessAuthentication>
        <tp:Endpoint
          tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/async"
          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:TransportServerSecurity>
      </tp:TransportReceiver>
```

```
      </tp:Transport>
      <tp:Transport tp:transportId="transportA2">
        <tp:TransportSender>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:TransportClientSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
            <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
          </tp:TransportClientSecurity>
        </tp:TransportSender>
        <tp:TransportReceiver>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</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:TransportServerSecurity>
        </tp:TransportReceiver>
      </tp:Transport>
      <tp:DocExchange tp:docExchangeId="docExchangeA1">
        <tp:ebXMLSenderBinding tp:version="2.0">
          <tp:ReliableMessaging>
            <tp:Retries>3</tp:Retries>
            <tp:RetryInterval>PT2H</tp:RetryInterval>
            <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
          </tp:ReliableMessaging>
          <tp:PersistDuration>P1D</tp:PersistDuration>
          <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:ebXMLSenderBinding>
        <tp:ebXMLReceiverBinding tp:version="2.0">
          <tp:ReliableMessaging>
            <tp:Retries>3</tp:Retries>
            <tp:RetryInterval>PT2H</tp:RetryInterval>
            <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
          </tp:ReliableMessaging>
          <tp:PersistDuration>P1D</tp:PersistDuration>
          <tp:ReceiverNonRepudiation>

<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:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
          </tp:ReceiverNonRepudiation>
          <tp:ReceiverDigitalEnvelope>
            <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
            <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
            <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
          </tp:ReceiverDigitalEnvelope>
        </tp:ebXMLReceiverBinding>
      </tp:DocExchange>
    </tp:PartyInfo>
    <!-- Party info for CompanyB -->
    <tp:PartyInfo
      tp:partyName="CompanyB"
      tp:defaultMshChannelId="asyncChannelB1"
      tp:defaultMshPackageId="CompanyB_MshSignalPackage">
      <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
      <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html"/>
      <tp:CollaborationRole>
        <tp:ProcessSpecification
```

```
        tp:version="2.0"
        tp:name="PIP3A4RequestPurchaseOrder"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml"
        tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
      <tp:Role
        tp:name="Seller"
        xlink:type="simple"
        xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
      <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert"/>
      <tp:ServiceBinding>
        <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID1"
            tp:action="Purchase Order Confirmation Action"
            tp:packageId="CompanyB_ResponsePackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT2H"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyA_ABID3</tp:OtherPartyActionBinding>
        </tp:CanSend>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID2"
            tp:action="ReceiptAcknowledgement"
            tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyA_ABID4</tp:OtherPartyActionBinding>
        </tp:CanSend>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID3"
            tp:action="Exception"
            tp:packageId="CompanyB_ExceptionPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"/>
            <tp:ChannelId>asyncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyA_ABID5</tp:OtherPartyActionBinding>
        </tp:CanSend>
        <tp:CanReceive>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID4"
            tp:action="Purchase Order Request Action"
            tp:packageId="CompanyB_RequestPackage">
            <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"/>
          <tp:ChannelId>asyncChannelB1</tp:ChannelId>
        </tp:ThisPartyActionBinding>
        <tp:OtherPartyActionBinding>companyA_ABID1</tp:OtherPartyActionBinding>
      </tp:CanReceive>
      <tp:CanReceive>
        <tp:ThisPartyActionBinding
          tp:id="companyB_ABID5"
          tp:action="ReceiptAcknowledgment"
          tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
          <tp:BusinessTransactionCharacteristics
            tp:isNonRepudiationRequired="true"
            tp:isNonRepudiationReceiptRequired="true"
            tp:isConfidential="transient"
            tp:isAuthenticated="persistent"
            tp:isTamperProof="persistent"
            tp:isAuthorizationRequired="true"/>
          <tp:ChannelId>asyncChannelB1</tp:ChannelId>
        </tp:ThisPartyActionBinding>
        <tp:OtherPartyActionBinding>companyA_ABID2</tp:OtherPartyActionBinding>
      </tp:CanReceive>
      <!-- The next binding uses a synchronous delivery channel -->
      <tp:CanReceive>
        <tp:ThisPartyActionBinding
          tp:id="companyB_ABID6"
          tp:action="Purchase Order Request Action"
          tp:packageId="CompanyB_RequestPackage">
          <tp:BusinessTransactionCharacteristics
            tp:isNonRepudiationRequired="true"
            tp:isNonRepudiationReceiptRequired="true"
            tp:isConfidential="transient"
            tp:isAuthenticated="persistent"
            tp:isTamperProof="persistent"
            tp:isAuthorizationRequired="true"
            tp:timeToAcknowledgeReceipt="PT5M"
            tp:timeToPerform="PT5M"/>
          <tp:ActionContext
            tp:binaryCollaboration="Request Purchase Order"
            tp:businessTransactionActivity="Request Purchase Order"
            tp:requestOrResponseAction="Purchase Order Request Action"/>
          <tp:ChannelId>syncChannelB1</tp:ChannelId>
        </tp:ThisPartyActionBinding>
        <tp:OtherPartyActionBinding>companyA_ABID6</tp:OtherPartyActionBinding>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID7"
            tp:action="Purchase Order Confirmation Action"
            tp:packageId="CompanyB_SyncReplyPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
              tp:isAuthenticated="persistent"
              tp:isTamperProof="persistent"
              tp:isAuthorizationRequired="true"
              tp:timeToAcknowledgeReceipt="PT5M"/>
            <tp:ActionContext
              tp:binaryCollaboration="Request Purchase Order"
              tp:businessTransactionActivity="Request Purchase Order"
              tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
            <tp:ChannelId>syncChannelB1</tp:ChannelId>
          </tp:ThisPartyActionBinding>
          <tp:OtherPartyActionBinding>companyA_ABID7</tp:OtherPartyActionBinding>
        </tp:CanSend>
        <tp:CanSend>
          <tp:ThisPartyActionBinding
            tp:id="companyB_ABID8"
            tp:action="Exception"
            tp:packageId="CompanyB_ExceptionPackage">
            <tp:BusinessTransactionCharacteristics
              tp:isNonRepudiationRequired="true"
              tp:isNonRepudiationReceiptRequired="true"
              tp:isConfidential="transient"
```

```
                tp:isAuthenticated="persistent"
                tp:isTamperProof="persistent"
                tp:isAuthorizationRequired="true"/>
              <tp:ChannelId>syncChannelB1</tp:ChannelId>
            </tp:ThisPartyActionBinding>
            <tp:OtherPartyActionBinding>companyA_ABID8</tp:OtherPartyActionBinding>
          </tp:CanSend>
        </tp:CanReceive>
      </tp:ServiceBinding>
    </tp:CollaborationRole>
    <!-- Certificates used by the "Seller" company -->
    <tp:Certificate tp:certId="CompanyB_AppCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_SigningCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_EncryptionCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_ServerCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="CompanyB_ClientCert">
      <ds:KeyInfo>
        <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB4">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB5">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB6">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB7">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:Certificate tp:certId="TrustedRootCertB8">
      <ds:KeyInfo>
        <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
      </ds:KeyInfo>
    </tp:Certificate>
    <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
      <tp:TrustAnchors>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8"/>
        <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7"/>
      </tp:TrustAnchors>
    </tp:SecurityDetails>
    <!-- An asynchronous delivery channel -->
    <tp:DeliveryChannel
      tp:channelId="asyncChannelB1"
      tp:transportId="transportB1"
```

```
        tp:docExchangeId="docExchangeB1">
        <tp:MessagingCharacteristics
          tp:syncReplyMode="none"
          tp:ackRequested="always"
          tp:ackSignatureRequested="always"
          tp:duplicateElimination="always"/>
      </tp:DeliveryChannel>
      <!-- A synchronous delivery channel -->
      <tp:DeliveryChannel
        tp:channelId="syncChannelB1"
        tp:transportId="transportB2"
        tp:docExchangeId="docExchangeB1">
        <tp:MessagingCharacteristics
          tp:syncReplyMode="signalsAndResponse"
          tp:ackRequested="always"
          tp:ackSignatureRequested="always"
          tp:duplicateElimination="always"/>
      </tp:DeliveryChannel>
      <tp:Transport tp:transportId="transportB1">
        <tp:TransportSender>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:TransportClientSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
            <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportClientSecurity>
        </tp:TransportSender>
        <tp:TransportReceiver>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:Endpoint
            tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
            tp:type="allPurpose"/>
          <tp:TransportServerSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
            <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportServerSecurity>
        </tp:TransportReceiver>
      </tp:Transport>
      <tp:Transport tp:transportId="transportB2">
        <tp:TransportSender>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:TransportClientSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
            <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportClientSecurity>
        </tp:TransportSender>
        <tp:TransportReceiver>
          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
          <tp:Endpoint
            tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
            tp:type="allPurpose"/>
          <tp:TransportServerSecurity>
            <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
            <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
            <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
          </tp:TransportServerSecurity>
        </tp:TransportReceiver>
      </tp:Transport>
      <tp:DocExchange tp:docExchangeId="docExchangeB1">
        <tp:ebXMLSenderBinding tp:version="2.0">
          <tp:ReliableMessaging>
            <tp:Retries>3</tp:Retries>
            <tp:RetryInterval>PT2H</tp:RetryInterval>
            <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
          </tp:ReliableMessaging>
          <tp:PersistDuration>P1D</tp:PersistDuration>
          <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="CompanyB_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="CompanyB_MessageSecurity"/>
        </tp:SenderDigitalEnvelope>
      </tp:ebXMLSenderBinding>
      <tp:ebXMLReceiverBinding tp:version="2.0">
        <tp:ReliableMessaging>
          <tp:Retries>3</tp:Retries>
          <tp:RetryInterval>PT2H</tp:RetryInterval>
          <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
        </tp:ReliableMessaging>
        <tp:PersistDuration>P1D</tp:PersistDuration>
        <tp:ReceiverNonRepudiation>

<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:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
        </tp:ReceiverNonRepudiation>
        <tp:ReceiverDigitalEnvelope>
          <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
          <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
          <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert"/>
        </tp:ReceiverDigitalEnvelope>
      </tp:ebXMLReceiverBinding>
    </tp:DocExchange>
  </tp:PartyInfo>
  <!-- SimplePart corresponding to the SOAP Envelope -->
  <tp:SimplePart
    tp:id="CompanyA_MsgHdr"
    tp:mimetype="text/xml">
    <tp:NamespaceSupported
      tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
      tp:version="2.0">
      http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
  </tp:NamespaceSupported>
  </tp:SimplePart>
  <tp:SimplePart
    tp:id="CompanyB_MsgHdr"
    tp:mimetype="text/xml">
    <tp:NamespaceSupported
      tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
      tp:version="2.0">
      http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
  </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
  <tp:SimplePart
    tp:id="CompanyA_ReceiptAcknowledgment"
    tp:mimetype="application/xml">
    <tp:NamespaceSupported
      tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
      tp:version="2.0">http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <tp:SimplePart
    tp:id="CompanyB_ReceiptAcknowledgment"
    tp:mimetype="application/xml">
    <tp:NamespaceSupported
      tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
      tp:version="2.0">
      http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
  <!-- SimplePart corresponding to an Exception business signal -->
  <tp:SimplePart
    tp:id="CompanyA_Exception"
    tp:mimetype="application/xml">
    <tp:NamespaceSupported
      tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
      tp:version="2.0">
      http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
    </tp:NamespaceSupported>
  </tp:SimplePart>
```

```
    <tp:SimplePart
      tp:id="CompanyB_Exception"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
        tp:version="2.0">
        http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <!-- SimplePart corresponding to a request action -->
    <tp:SimplePart
      tp:id="CompanyA_Request"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
        tp:version="1.0">
        http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <tp:SimplePart
      tp:id="CompanyB_Request"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
        tp:version="1.0">
        http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <!-- SimplePart corresponding to a response action -->
    <tp:SimplePart
      tp:id="CompanyA_Response"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
        tp:version="1.0">
        http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <tp:SimplePart
      tp:id="CompanyB_Response"
      tp:mimetype="application/xml">
      <tp:NamespaceSupported
        tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
        tp:version="1.0">
        http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
      </tp:NamespaceSupported>
    </tp:SimplePart>
    <!-- An ebXML message with a SOAP Envelope only -->
    <tp:Packaging
      tp:id="CompanyA_MshSignalPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_MshSignal"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Packaging
      tp:id="CompanyB_MshSignalPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_MshSignal"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
    <tp:Packaging tp:id="CompanyA_RequestPackage">
```

```
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_RequestMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_Request"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Packaging tp:id="CompanyB_RequestPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_RequestMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_Request"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
    <tp:Packaging tp:id="CompanyA_ResponsePackage">
      <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_ResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_Response"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Packaging tp:id="CompanyB_ResponsePackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_ResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_Response"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
    <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_ReceiptAcknowledgmentMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_ReceiptAcknowledgmentMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
```

```
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
    <tp:Packaging tp:id="CompanyA_ExceptionPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_ExceptionMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_Exception"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Packaging tp:id="CompanyB_ExceptionPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_ExceptionMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_Exception"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
         or an ebXML message with an Exception signal -->
    <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyA_SignalAndResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
          <tp:Constituent tp:idref="CompanyA_Response"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
      <tp:ProcessingCapabilities
        tp:parse="true"
        tp:generate="true"/>
      <tp:CompositeList>
        <tp:Composite
          tp:id="CompanyB_SignalAndResponseMsg"
          tp:mimetype="multipart/related"
          tp:mimeparameters="type=text/xml">
          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
          <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
          <tp:Constituent tp:idref="CompanyB_Response"/>
        </tp:Composite>
      </tp:CompositeList>
    </tp:Packaging>
    <tp:Comment xml:lang="en-US">buy/sell agreement between CompanyA.com and
  CompanyB.com</tp:Comment>
  </tp:CollaborationProtocolAgreement>
```

Appendix C Business Process Specification Corresponding to Complete CPP and CPA Definition (Non-Normative)

This Business Process Specification referenced by the CPPs and CPA in Appendix A and Appendix B are reproduced here. This document is available as an ASCII file at:

http://www.oasis-open.org/committees/ebxml-cppa/schema/bpss-example-2_0a.xml

The schema to which this instance document conforms is available as an ASCII file at:

http://www.oasis-open.org/committees/ebxml-cppa/schema/ebBPSS1.04.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessSpecification
  xmlns="http://www.ebxml.org/BusinessProcess"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ebxml.org/BusinessProcess ebBPSS1.04.xsd"
  name="PIP3A4RequestPurchaseOrder"
  uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"
  version="R02.00">
  <Documentation>
    This PIP enables a buyer to issue a purchase order and obtain a quick response
    from the provider that acknowledges which of the purchase order product line
    items are accepted, rejected, or pending.
  </Documentation>
  <!--Purchase order Request Document-->
  <BusinessDocument
    name="Puchase Order Request"
    nameID="Pip3A4PurchaseOrderRequest"
    specificationLocation="PurchaseOrderRequest.xsd">
    <Documentation>
      The document is an XSD file that specifies the rules for creating the XML
      document for the business action of requesting a purchase order.
    </Documentation>
  </BusinessDocument>
  <BusinessDocument
    name="Puchase Order Confirmation"
    nameID="Pip3A4PurchaseOrderConfirmation"
    specificationLocation="PurchaseOrderConfirmation.xsd">
    <Documentation>
      The document is an XSD file that specifies the rules for creating the XML
      document for the business action of making a purchase order confirmation.
    </Documentation>
  </BusinessDocument>
  <BusinessTransaction
    name="Request Purchase Order"
    nameID="RequestPurchaseOrder_BT">
    <RequestingBusinessActivity
      name="Purchase Order Request Action"
      nameID="PurchaseOrderRequestAction"
      isAuthorizationRequired ="true"
      isIntelligibleCheckRequired="true"
      isNonRepudiationReceiptRequired="true"
      isNonRepudiationRequired="true"
      timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
      <DocumentEnvelope
        businessDocument="Puchase Order Request"
        businessDocumentIDRef="Pip3A4PurchaseOrderRequest"
        isAuthenticated="persistent"
        isConfidential="transient"
        isTamperProof="persistent"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity
      name="Purchase Order Confirmation Action"
      nameID="PurchaseOrderConfirmationAction"
      isAuthorizationRequired="true"
      isIntelligibleCheckRequired="true"
      isNonRepudiationRequired="true"
      timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
      <DocumentEnvelope
```

```
          businessDocument="Purchase Order Confirmation"
          businessDocumentIDRef="Pip3A4PurchaseOrderConfirmation"
          isAuthenticated="persistent"
          isConfidential="transient"
          isPositiveResponse="true"
          isTamperProof="persistent"/>
      </RespondingBusinessActivity>
    </BusinessTransaction>
    <BinaryCollaboration
      name="Request Purchase Order"
      nameID="RequestPurchaseOrder_BC"
      initiatingRole="BuyerId">
      <Role
        name="Buyer"
        nameID="BuyerId"/>
      <Role
        name="Seller"
        nameID="SellerId"/>
      <Start toBusinessState="Request Purchase Order"/>
      <BusinessTransactionActivity
        name="Request Purchase Order"
        nameID="RequestPurchaseOrder_BTA"
        businessTransaction="Request Purchase Order"
        businessTransactionIDRef="RequestPurchaseOrder_BT"
        fromRole="Buyer" fromRoleIDRef="BuyerId"
        toRole="Seller" toRoleIDRef="SellerId"
        isLegallyBinding="true"
        timeToPerform="P0Y0M0DT24H0M0S"
        isConcurrent="false"/>
      <Success
        fromBusinessState="Request Purchase Order"
        conditionGuard="Success"/>
      <Failure
        fromBusinessState="Request Purchase Order"
        conditionGuard="BusinessFailure"/>
      <Transition
        fromBusinessState="Request Purchase Order"
        toBusinessState="Request Purchase Order"/>
    </BinaryCollaboration>
  </ProcessSpecification>
```

Appendix D W3C XML Schema Document Corresponding to Complete CPP and CPA Definition (Normative)

This XML Schema document is available as an ASCII file at:
http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0b.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is the schema that corresponds to the version 2.0 CPP/A spec -->
<!-- Some parsers may require explicit declaration of
'xmlns:xml="http://www.w3.org/XML/1998/namespace"'.
     In that case, a copy of this schema augmented with the above declaration should be cached
and used
     for the purpose of schema validation for CPPs and CPAs. -->
<schema
  targetNamespace="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
  xmlns:tns="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="qualified"
  attributeFormDefault="qualified" version="2_0b">
  <import
    namespace="http://www.w3.org/1999/xlink"
    schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/xlink.xsd"/>
  <import
    namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
  <import
    namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
  <attributeGroup name="pkg.grp">
    <attribute ref="tns:id" use="required"/>
    <attribute name="mimetype" type="tns:non-empty-string" use="required"/>
    <attribute name="mimeparameters" type="tns:non-empty-string"/>
  </attributeGroup>
  <attributeGroup name="xlink.grp">
    <attribute ref="xlink:type" fixed="simple"/>
    <attribute ref="xlink:href" use="required"/>
  </attributeGroup>
  <element name="CollaborationProtocolAgreement">
    <complexType>
      <sequence>
        <element ref="tns:Status"/>
        <element ref="tns:Start"/>
        <element ref="tns:End"/>
        <element ref="tns:ConversationConstraints" minOccurs="0"/>
        <element ref="tns:PartyInfo" minOccurs="2" maxOccurs="2"/>
        <element ref="tns:SimplePart" maxOccurs="unbounded"/>
        <element ref="tns:Packaging" maxOccurs="unbounded"/>
        <element ref="tns:Signature" minOccurs="0"/>
        <element ref="tns:Comment" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="cpaid" type="tns:non-empty-string" use="required"/>
      <attribute ref="tns:version" use="required"/>
    </complexType>
  </element>
  <element name="Signature">
    <complexType>
      <sequence>
        <element ref="ds:Signature" maxOccurs="3"/>
      </sequence>
    </complexType>
  </element>
  <element name="CollaborationProtocolProfile">
    <complexType>
      <sequence>
        <element ref="tns:PartyInfo" maxOccurs="unbounded"/>
        <element ref="tns:SimplePart" maxOccurs="unbounded"/>
        <element ref="tns:Packaging" maxOccurs="unbounded"/>
        <element ref="tns:Signature" minOccurs="0"/>
        <element ref="tns:Comment" minOccurs="0" maxOccurs="unbounded"/>
```

```
        </sequence>
        <attribute name="cppid" type="tns:non-empty-string" use="required"/>
        <attribute ref="tns:version" use="required"/>
      </complexType>
    </element>
    <element name="ProcessSpecification">
      <complexType>
        <sequence>
          <element ref="ds:Reference" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="name" type="tns:non-empty-string" use="required"/>
        <attribute ref="tns:version" use="required"/>
        <attributeGroup ref="tns:xlink.grp"/>
        <attribute name="uuid" type="anyURI"/>
      </complexType>
    </element>
    <element name="Service" type="tns:service.type"/>
    <element name="Protocol" type="tns:protocol.type"/>
    <element name="SendingProtocol" type="tns:protocol.type"/>
    <element name="ReceivingProtocol" type="tns:protocol.type"/>
    <element name="OverrideMshActionBinding">
      <complexType>
        <attribute name="action" type="tns:non-empty-string" use="required"/>
        <attribute name="channelId" type="IDREF" use="required"/>
      </complexType>
    </element>
    <element name="ChannelId" type="IDREF"/>
    <complexType name="ActionBinding.type">
      <sequence>
        <element ref="tns:BusinessTransactionCharacteristics"/>
        <element ref="tns:ActionContext" minOccurs="0"/>
        <element ref="tns:ChannelId" maxOccurs="unbounded"/>
        <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="id" type="ID" use="required"/>
      <attribute name="action" type="tns:non-empty-string" use="required"/>
      <attribute name="packageId" type="IDREF" use="required"/>
      <attribute ref="xlink:href" use="optional"/>
      <attribute ref="xlink:type" fixed="simple"/>
    </complexType>
    <element name="ActionContext">
      <complexType>
        <sequence>
          <element ref="tns:CollaborationActivity" minOccurs="0"/>
          <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="binaryCollaboration" type="tns:non-empty-string" use="required"/>
        <attribute name="businessTransactionActivity" type="tns:non-empty-string" use="required"/>
        <attribute name="requestOrResponseAction" type="tns:non-empty-string" use="required"/>
      </complexType>
    </element>
    <element name="CollaborationActivity">
      <complexType>
        <sequence>
          <element ref="tns:CollaborationActivity" minOccurs="0"/>
        </sequence>
        <attribute name="name" type="tns:non-empty-string"/>
      </complexType>
    </element>
    <element name="CollaborationRole">
      <complexType>
        <sequence>
          <element ref="tns:ProcessSpecification"/>
          <element ref="tns:Role"/>
          <element name="ApplicationCertificateRef" type="tns:CertificateRef.type" minOccurs="0"
maxOccurs="unbounded"/>
          <element name="ApplicationSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
minOccurs="0"/>
          <element ref="tns:ServiceBinding"/>
        </sequence>
      </complexType>
    </element>
    <element name="PartyInfo">
      <complexType>
        <sequence>
          <element ref="tns:PartyId" maxOccurs="unbounded"/>
          <element ref="tns:PartyRef" maxOccurs="unbounded"/>
          <element ref="tns:CollaborationRole" maxOccurs="unbounded"/>
```

```
        <element ref="tns:Certificate" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:SecurityDetails" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:DeliveryChannel" maxOccurs="unbounded"/>
        <element ref="tns:Transport" maxOccurs="unbounded"/>
        <element ref="tns:DocExchange" maxOccurs="unbounded"/>
        <element ref="tns:OverrideMshActionBinding" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="partyName" type="tns:non-empty-string" use="required"/>
      <attribute name="defaultMshChannelId" type="IDREF" use="required"/>
      <attribute name="defaultMshPackageId" type="IDREF" use="required"/>
    </complexType>
  </element>
  <element name="PartyId">
    <complexType>
      <simpleContent>
        <extension base="tns:non-empty-string">
          <attribute name="type" type="anyURI"/>
        </extension>
      </simpleContent>
    </complexType>
  </element>
  <element name="PartyRef">
    <complexType>
      <sequence>
      </sequence>
      <attributeGroup ref="tns:xlink.grp"/>
      <attribute name="type" type="anyURI"/>
      <attribute name="schemaLocation" type="anyURI"/>
    </complexType>
  </element>
  <element name="DeliveryChannel">
    <complexType>
      <sequence>
        <element ref="tns:MessagingCharacteristics"/>
      </sequence>
      <attribute name="channelId" type="ID" use="required"/>
      <attribute name="transportId" type="IDREF" use="required"/>
      <attribute name="docExchangeId" type="IDREF" use="required"/>
    </complexType>
  </element>
  <element name="Transport">
    <complexType>
      <sequence>
        <element ref="tns:TransportSender" minOccurs="0"/>
        <element ref="tns:TransportReceiver" minOccurs="0"/>
      </sequence>
      <attribute name="transportId" type="ID" use="required"/>
    </complexType>
  </element>
  <element name="AccessAuthentication" type="tns:accessAuthentication.type"/>
  <element name="TransportSender">
    <complexType>
      <sequence>
        <element name="TransportProtocol" type="tns:protocol.type"/>
        <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:TransportClientSecurity" minOccurs="0"/>
      </sequence>
    </complexType>
  </element>
  <element name="TransportReceiver">
    <complexType>
      <sequence>
        <element name="TransportProtocol" type="tns:protocol.type"/>
        <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:Endpoint" maxOccurs="unbounded"/>
        <element ref="tns:TransportServerSecurity" minOccurs="0"/>
      </sequence>
    </complexType>
  </element>
  <element name="Endpoint">
    <complexType>
      <attribute name="uri" type="anyURI" use="required"/>
      <attribute name="type" type="tns:endpointType.type" default="allPurpose"/>
    </complexType>
  </element>
  <element name="TransportClientSecurity">
    <complexType>
      <sequence>
```

```
        <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
        <element name="ClientCertificateRef" type="tns:CertificateRef.type" minOccurs="0"/>
        <element name="ServerSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
minOccurs="0"/>
        <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
  <element name="TransportServerSecurity">
    <complexType>
      <sequence>
        <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
        <element name="ServerCertificateRef" type="tns:CertificateRef.type"/>
        <element name="ClientSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
minOccurs="0"/>
        <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
  <element name="Certificate">
    <complexType>
      <sequence>
        <element ref="ds:KeyInfo"/>
      </sequence>
      <attribute name="certId" type="ID" use="required"/>
    </complexType>
  </element>
  <element name="DocExchange">
    <complexType>
      <sequence>
        <element ref="tns:ebXMLSenderBinding" minOccurs="0"/>
        <element ref="tns:ebXMLReceiverBinding" minOccurs="0"/>
      </sequence>
      <attribute name="docExchangeId" type="ID" use="required"/>
    </complexType>
  </element>
  <element name="ReliableMessaging">
    <complexType>
      <sequence>
        <element name="Retries" type="integer" minOccurs="0"/>
        <element name="RetryInterval" type="duration" minOccurs="0"/>
        <element name="MessageOrderSemantics" type="tns:messageOrderSemantics.type"/>
      </sequence>
    </complexType>
  </element>
  <element name="PersistDuration" type="duration"/>
  <element name="SenderNonRepudiation">
    <complexType>
      <sequence>
        <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
        <element ref="tns:HashFunction"/>
        <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>
        <element name="SigningCertificateRef" type="tns:CertificateRef.type"/>
      </sequence>
    </complexType>
  </element>
  <element name="ReceiverNonRepudiation">
    <complexType>
      <sequence>
        <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
        <element ref="tns:HashFunction"/>
        <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>
        <element name="SigningSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
minOccurs="0"/>
      </sequence>
    </complexType>
  </element>
  <element name="HashFunction" type="tns:non-empty-string"/>
  <element name="EncryptionAlgorithm">
    <complexType>
      <simpleContent>
        <extension base="tns:non-empty-string">
          <attribute name="minimumStrength" type="integer"/>
          <attribute name="oid" type="tns:non-empty-string"/>
          <attribute name="w3c" type="tns:non-empty-string"/>
          <attribute name="enumerationType" type="tns:non-empty-string"/>
        </extension>
      </simpleContent>
```

```
      </complexType>
    </element>
    <element name="SignatureAlgorithm">
      <complexType>
        <simpleContent>
          <extension base="tns:non-empty-string">
            <attribute name="oid" type="tns:non-empty-string"/>
            <attribute name="w3c" type="tns:non-empty-string"/>
            <attribute name="enumerationType" type="tns:non-empty-string"/>
          </extension>
        </simpleContent>
      </complexType>
    </element>
    <element name="SenderDigitalEnvelope">
      <complexType>
        <sequence>
          <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
          <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
          <element name="EncryptionSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
minOccurs="0"/>
        </sequence>
      </complexType>
    </element>
    <element name="ReceiverDigitalEnvelope">
      <complexType>
        <sequence>
          <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
          <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
          <element name="EncryptionCertificateRef" type="tns:CertificateRef.type"/>
        </sequence>
      </complexType>
    </element>
    <element name="ebXMLSenderBinding">
      <complexType>
        <sequence>
          <element ref="tns:ReliableMessaging" minOccurs="0"/>
          <element ref="tns:PersistDuration" minOccurs="0"/>
          <element ref="tns:SenderNonRepudiation" minOccurs="0"/>
          <element ref="tns:SenderDigitalEnvelope" minOccurs="0"/>
          <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute ref="tns:version" use="required"/>
      </complexType>
    </element>
    <element name="ebXMLReceiverBinding">
      <complexType>
        <sequence>
          <element ref="tns:ReliableMessaging" minOccurs="0"/>
          <element ref="tns:PersistDuration" minOccurs="0"/>
          <element ref="tns:ReceiverNonRepudiation" minOccurs="0"/>
          <element ref="tns:ReceiverDigitalEnvelope" minOccurs="0"/>
          <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute ref="tns:version" use="required"/>
      </complexType>
    </element>
    <element name="NamespaceSupported">
      <complexType>
        <simpleContent>
          <extension base="anyURI">
            <attribute name="location" type="anyURI" use="required"/>
            <attribute ref="tns:version"/>
          </extension>
        </simpleContent>
      </complexType>
    </element>
    <element name="BusinessTransactionCharacteristics">
      <complexType>
        <attribute name="isNonRepudiationRequired" type="boolean"/>
        <attribute name="isNonRepudiationReceiptRequired" type="boolean"/>
        <attribute name="isConfidential" type="tns:persistenceLevel.type"/>
        <attribute name="isAuthenticated" type="tns:persistenceLevel.type"/>
        <attribute name="isTamperProof" type="tns:persistenceLevel.type"/>
        <attribute name="isAuthorizationRequired" type="boolean"/>
        <attribute name="isIntelligibleCheckRequired" type="boolean"/>
        <attribute name="timeToAcknowledgeReceipt" type="duration"/>
        <attribute name="timeToAcknowledgeAcceptance" type="duration"/>
        <attribute name="timeToPerform" type="duration"/>
```

```
        <attribute name="retryCount" type="integer"/>
      </complexType>
    </element>
    <element name="MessagingCharacteristics">
      <complexType>
        <attribute ref="tns:syncReplyMode" default="none"/>
        <attribute name="ackRequested" type="tns:perMessageCharacteristics.type"
default="perMessage"/>
        <attribute name="ackSignatureRequested" type="tns:perMessageCharacteristics.type"
default="perMessage"/>
        <attribute name="duplicateElimination" type="tns:perMessageCharacteristics.type"
default="perMessage"/>
        <attribute name="actor" type="tns:actor.type"/>
      </complexType>
    </element>
    <element name="ServiceBinding">
      <complexType>
        <sequence>
          <element ref="tns:Service"/>
          <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded"/>
          <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="CanSend">
      <complexType>
        <sequence>
          <element name="ThisPartyActionBinding" type="tns:ActionBinding.type"/>
          <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0"/>
          <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="CanReceive">
      <complexType>
        <sequence>
          <element name="ThisPartyActionBinding" type="tns:ActionBinding.type"/>
          <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0"/>
          <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="Status">
      <complexType>
        <attribute name="value" type="tns:statusValue.type" use="required"/>
      </complexType>
    </element>
    <element name="Start" type="dateTime"/>
    <element name="End" type="dateTime"/>
    <element name="Type" type="tns:non-empty-string"/>
    <element name="ConversationConstraints">
      <complexType>
        <attribute name="invocationLimit" type="int"/>
        <attribute name="concurrentConversations" type="int"/>
      </complexType>
    </element>
    <element name="Role">
      <complexType>
        <attribute name="name" type="tns:non-empty-string" use="required"/>
        <attributeGroup ref="tns:xlink.grp"/>
      </complexType>
    </element>
    <element name="SignatureTransforms">
      <complexType>
        <sequence>
          <element ref="ds:Transform" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="EncryptionTransforms">
      <complexType>
        <sequence>
          <element ref="ds:Transform" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="Constituent">
      <complexType>
```

```
      <sequence>
        <element ref="tns:SignatureTransforms" minOccurs="0"/>
        <element ref="tns:EncryptionTransforms" minOccurs="0"/>
      </sequence>
      <attribute ref="tns:idref" use="required"/>
      <attribute name="excludedFromSignature" type="boolean" default="false"/>
      <attribute name="minOccurs" type="nonNegativeInteger"/>
      <attribute name="maxOccurs" type="nonNegativeInteger"/>
    </complexType>
  </element>
  <element name="Packaging">
    <complexType>
      <sequence>
        <element name="ProcessingCapabilities">
          <complexType>
            <attribute name="parse" type="boolean" use="required"/>
            <attribute name="generate" type="boolean" use="required"/>
          </complexType>
        </element>
        <element name="CompositeList" maxOccurs="unbounded">
          <complexType>
            <choice maxOccurs="unbounded">
              <element name="Encapsulation">
                <complexType>
                  <sequence>
                    <element ref="tns:Constituent"/>
                  </sequence>
                  <attributeGroup ref="tns:pkg.grp"/>
                </complexType>
              </element>
              <element name="Composite">
                <complexType>
                  <sequence>
                    <element ref="tns:Constituent" maxOccurs="unbounded"/>
                  </sequence>
                  <attributeGroup ref="tns:pkg.grp"/>
                </complexType>
              </element>
            </choice>
          </complexType>
        </element>
      </sequence>
      <attribute ref="tns:id" use="required"/>
    </complexType>
  </element>
  <element name="Comment">
    <complexType>
      <simpleContent>
        <extension base="tns:non-empty-string">
          <attribute ref="xml:lang"/>
        </extension>
      </simpleContent>
    </complexType>
  </element>
  <element name="SimplePart">
    <complexType>
      <sequence>
        <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attributeGroup ref="tns:pkg.grp"/>
      <attribute ref="xlink:role"/>
    </complexType>
  </element>
  <!-- COMMON -->
  <simpleType name="statusValue.type">
    <restriction base="NMTOKEN">
      <enumeration value="agreed"/>
      <enumeration value="signed"/>
      <enumeration value="proposed"/>
    </restriction>
  </simpleType>
  <simpleType name="endpointType.type">
    <restriction base="NMTOKEN">
      <enumeration value="login"/>
      <enumeration value="request"/>
      <enumeration value="response"/>
      <enumeration value="error"/>
      <enumeration value="allPurpose"/>
```

```
      </restriction>
    </simpleType>
    <simpleType name="non-empty-string">
      <restriction base="string">
        <minLength value="1"/>
      </restriction>
    </simpleType>
    <simpleType name="syncReplyMode.type">
      <restriction base="NMTOKEN">
        <enumeration value="mshSignalsOnly"/>
        <enumeration value="responseOnly"/>
        <enumeration value="signalsAndResponse"/>
        <enumeration value="signalsOnly"/>
        <enumeration value="none"/>
      </restriction>
    </simpleType>
    <complexType name="service.type">
      <simpleContent>
        <extension base="tns:non-empty-string">
          <attribute name="type" type="tns:non-empty-string"/>
        </extension>
      </simpleContent>
    </complexType>
    <complexType name="protocol.type">
      <simpleContent>
        <extension base="tns:non-empty-string">
          <attribute ref="tns:version"/>
        </extension>
      </simpleContent>
    </complexType>
    <attribute name="idref" type="IDREF"/>
    <attribute name="id" type="ID"/>
    <attribute name="version" type="tns:non-empty-string"/>
    <attribute name="syncReplyMode" type="tns:syncReplyMode.type"/>
    <complexType name="SecurityPolicy.type"/>
    <complexType name="CertificateRef.type">
      <attribute name="certId" type="IDREF" use="required"/>
    </complexType>
    <simpleType name="perMessageCharacteristics.type">
      <restriction base="NMTOKEN">
        <enumeration value="always"/>
        <enumeration value="never"/>
        <enumeration value="perMessage"/>
      </restriction>
    </simpleType>
    <simpleType name="actor.type">
      <restriction base="NMTOKEN">
        <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:nextMSH"/>
        <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"/>
      </restriction>
    </simpleType>
    <simpleType name="messageOrderSemantics.type">
      <restriction base="Name">
        <enumeration value="Guaranteed"/>
        <enumeration value="NotGuaranteed"/>
      </restriction>
    </simpleType>
    <complexType name="SecurityDetailsRef.type">
      <attribute name="securityId" type="IDREF" use="required"/>
    </complexType>
    <simpleType name="persistenceLevel.type">
      <restriction base="Name">
        <enumeration value="none"/>
        <enumeration value="transient"/>
        <enumeration value="persistent"/>
        <enumeration value="transient-and-persistent"/>
      </restriction>
    </simpleType>
    <element name="SecurityDetailsRef" type="tns:SecurityDetailsRef.type"/>
    <element name="SecurityDetails">
      <complexType>
        <sequence>
          <element ref="tns:TrustAnchors" minOccurs="0"/>
          <element ref="tns:SecurityPolicy" minOccurs="0"/>
        </sequence>
        <attribute name="securityId" type="ID" use="required"/>
      </complexType>
    </element>
```

```
  <element name="TrustAnchors">
    <complexType>
      <sequence>
        <element name="AnchorCertificateRef" type="tns:CertificateRef.type"
maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
  <element name="SecurityPolicy">
    <complexType>
      <sequence>
      </sequence>
    </complexType>
  </element>
  <simpleType name="accessAuthentication.type">
    <restriction base="NMTOKEN">
      <enumeration value="basic"/>
      <enumeration value="digest"/>
    </restriction>
  </simpleType>
</schema>
```

Appendix E CPA Composition (Non-Normative)

E.1 Suggestions for Design of Computational Procedures

A quick inspection of the schemas for the top level elements, ***CollaborationProtocolProfile*** **(***CPP*) and ***CollaborationProtocolAgreement*** **(***CPA*)**,** shows that a *CPA* can be viewed as a result of merging portions of the ***PartyInfo*** elements found in constituent *CPP*s, and then integrating these ***PartyInfo*** elements with other *CPA* sibling elements, such as those governing the *CPA* validity period.

Merging *CPP*s into *CPA*s is one way in which trading partners can arrive at a proposed or "draft" *CPA*. A draft *CPA* might also be formed from a *CPA* template. A *CPA*-template represents one party's proposed implementation of a business process that uses placeholding values for the identifying aspects of the other party, such as ***PartyId*** or ***TransportEndpoint*** elements. To form a *CPA* from a *CPA* template, the placeholder values are replaced by the actual values for the other trading partner. The actual values could themselves be extracted from the other trading partner's *CPP***,** if one is available, or they could be obtained from an administrator performing data entry functions.

We call objects draft *CPA*s to indicate their potential use as inputs to a *CPA* negotiation process in which a draft *CPA* is verified as suitable for both *Parties*, modified until a suitable *CPA* is found, or discovered to not be feasible until one side (or both) acquires additional software capabilities. These negotiation procedures and protocols are currently being designed, their requirements having been defined, and the resulting specifications should be available with the next release of this specification. In general, a draft *CPA* will constitute a proposal about an overall binding of a business process to a delivery implementation, while negotiation will be used to arrive at detailed values for parameters reflecting a final agreement. A special companion document, the *NegotiationDescriptorDocument,* provides both focus on what parameters can be negotiated as well as ranges or sets of acceptable values for those parameters.

In the remainder of this appendix, the goal will be to identify and describe the basic tasks that computational procedures for the assembly of the draft *CPA* would normally accomplish. While no normative specification is provided for an algorithm for *CPA* formation, some guidance for implementers is provided. This information might assist the software implementer in designing a partially automated and partially interactive software system useful for configuring *Business Collaboration* so as to arrive at satisfactorily complete levels of interoperability.

Before enumerating and describing the basic tasks, it is worthwhile mentioning two basic reasons why we focus on the component tasks involved in *CPA* formation rather than attempt to provide an algorithm for *CPA* formation. These reasons provide some hints to implementers about ways in which they might customize their approaches to drafting *CPA*s from *CPP*s**.**

E.1.1 Variability in Inputs

User preferences provide one source of variability in the inputs to the *CPA* formation process. Let us suppose in this section that each of the *Parties* has made its *CPP* available to potential

collaborators. Normally one *Party* will have a desired *Business Collaboration* (defined in a ***ProcessSpecification*** document*)* to implement with its intended collaborator. So the information inputs will normally involve a user preference about intended *Business Collaborations* in addition to just the *CPPs.*

A *CPA* formation tool can have access to local user information not advertised in the *CPP* that can contribute to the *CPA* that is formed. A user can have chosen to only advertise those system capabilities that reflect capabilities that have not been deprecated. For example, a user can only advertise HTTP and omit FTP, even when capable of using FTP. The reason for omitting FTP might be concerns about the scalability of managing user accounts, directories, and passwords for FTP sessions. Despite not advertising an FTP capability, configuration software can use tacit knowledge about its own FTP capability to form a *CPA* with an intended collaborator who happens to have only an FTP capability for implementing a desired *Business Collaboration*. In other words, business interests can, in this case, override the deprecation policy. Both tacit knowledge and detailed preference information account for variability in inputs into the *CPA* formation process.

E.1.2 Variable Stringency in Evaluating Proposed Agreements

The conditions for output of a *CPA* given two *CPPs* can involve different levels and extents of interoperability. In other words, when an optimal solution that satisfies every level of requirement and every other additional constraint does not exist, a *Party* can propose a *CPA* that satisfies enough of the requirements for "a good enough" implementation. User input can be solicited to determine what is a good enough implementation, and so can be as varied as there are user configuration options to express preferences. In practice, compromises can be made on security, reliable messaging, levels of signals and acknowledgments, and other matters in order to find some acceptable means of doing business.

A *CPA* can support a fully interoperable configuration in which agreement has been reached on all technical levels needed for *Business Collaboration*. In such a case, matches in capabilities will have been found in all relevant technical levels.

However, there can be interoperable configurations agreed to in a *CPA* in which not all aspects of a *Business Collaboration* match. Gaps can exist in packaging, security, signaling, reliable messaging and other areas and yet the systems can still transport the business data, and special means can be employed to handle the exceptions. In such situations, a *CPA* can reflect configured policies or expressly solicited user permission to ignore some shortcomings in configurations. A system might not be capable of responding in a *Business Collaboration* so as to support a specified ability to supply non-repudiation of receipt, but might still be acceptable for business reasons. A system might not be able to handle all the processing needed to support, for example, SOAP with Attachments and yet still be able to treat the multipart according to "multipart/mixed" handling and allow *Business Collaboration* to take place. In fact, short of a failure to be able to transport data and a failure to be able to provide data relevant to the *Business Collaboration*, there are few features that might not be temporarily or indefinitely compromised about, given overriding *business* interests. This situation of "partial interoperability" is to be

expected to persist for some time, and so interferes with formulating a "clean" algorithm for deciding on what is sufficient for interoperability.

E.2 CPA Formation Component Tasks

Technically viewed, a *CPA* provides "bindings" between *Business Collaboration* specifications (such as those defined within the ***ProcessSpecification***'s referenced documents) and those services and protocols that are used to implement these specifications. The implementation takes place at several levels and involves varied services at these levels. A *CPA* that arrives at a fully interoperable collaboration binding can be thought of as arriving at interoperable, application-to-application integration. *CPAs* can fall short of this goal and still be both useful and acceptable to the collaborating *Parties*. Certainly, if no matching data-transport capabilities can be discovered, a *CPA* would not provide much in the way of interoperable integration. Likewise, partial *CPAs* can leave significant system work to be done before a completely satisfactory application-to-application integration is realized. Even so, partial integration can be sufficient to allow collaboration, and to enjoy payoffs from increased levels of automation.

In practice, the *CPA* formation process can produce a complete *CPA*, a failure result, a gap list that drives a dialog with the user, or perhaps even a *CPA* that implements partial interoperability "good enough" for the business collaborators. Because both matching capabilities and interoperability can be matters of degree, the constituent tasks are finding the matches in capabilities at different levels and for different services. We next proceed to characterize the most important of these constituent tasks.

E.3 CPA Formation from *CPP*s: Context of Tasks

To simplify discussion, assume in the following that we are viewing the tasks faced by a software agent when:

1. an intended collaborator is known and the collaborator's *CPP* has been retrieved,
2. the ***ProcessSpecification*** between our side and our intended collaborator has been selected,
3. the ***Service, Action,*** and the specific ***Role*** elements that our software agent is to play in the *Business Collaboration* (with discussion soon restricted to ***BinaryCollaboration****s)* is known, and
4. finally, the capabilities that we have advertised in our *CPP* are known

For vividness, we will develop our discussions using the "3A4" ebBPSS example and the *CPP*s of Company A and B that are found in full in appendices of this document and that should also be available at the web site for the OASIS ebXML CPPA Technical Committee. For simplicity, we will assume that the information about capabilities is restricted to what is available in our agent's *CPP,* and in the *CPP* of our intended collaborator. We will suppose that we have taken on the viewpoint of Company A assembling a draft *CPA*. Please note that there is no guarantee that the same draft *CPA*s will be produced in the same order from differing viewpoints.

In general, the basic tasks consist of finding "matches" between our capabilities and our intended collaborator's capabilities at the various levels of the collaboration protocol stack and with respect to the services supplied at these various levels. This stack, which need not be characterized in any detail, is at least distinguished by an application level and a messaging transfer level. The application level is governed by a business process flow specification, such as ebBPSS. The messaging transfer level will consist of a number of requirements and options concerning transfer protocols, security, packaging, and messaging patterns (such as various kinds of acknowledgment, error messages, and the like.)

In actually assembling the tasks into a computational process, it will generally make sense to perform the tasks in a certain order. The overall order reflects the implicit structure of the *CPA:* first undertake those tasks to ensure that there is a match with respect to the *Business Collaboration* process. Without finding that the collaborators can participate in the same ***ProcessSpecification*** successfully, there is little point in working through implementation options. Then, examine the matches within the components of the bindings that have been announced for the *Business Collaboration* process, checking for the most indispensable "matches" first (***Transport***-related), and continuing checks on the other layers reflecting integrated interoperability at packaging, security, signals and protocol patterns, and so on. With this basic overview in mind, let us proceed to consider the basic tasks in greater detail.

E.4 Business Collaboration Process Matching Tasks

Company A has announced within its *CPP*, at least one ***PartyInfo*** element. For current purposes, the most important initial focus is on all the sibling elements with the path ***/CollaborationProtocolProfile/PartyInfo/CollaborationRole.*** Each element of this kind has a child, ***ProcessSpecification.*** Our initial matching task (probably better viewed as a filtering task) is to select those nodes where the ***ProcessSpecification*** is one that we are interested in building a CPA for! Checking the attribute values allows us to select by comparing values in the ***name***, ***xlink:href*** or ***uuid*** attributes. The definitive value for matching **ebBPSS** process specifications is the value found in the ***ProcessSpecification/@uuid*** attribute.

E.4.1 Matching *ProcessSpecification*/*Role*s, and *Actions:* Initial Filtering and Selection

The previous task has essentially found two ***CollaborationRole*** node sets within our and our collaborator's CPP documents where the ***ProcessSpecifications*** are identical, and equal to the value of interest given above. In other words, we have ***CollaborationRole***s with ***ProcessSpecification/@name=***'PIP3A4RequestPurchaseOrder'. It is convenient but not essential to use the ***name*** attribute in performing this selection.

We next proceed to filter these node sets. We have been given our ***Role*** element value for our participation in the ProcessSpecification. For Company A, this ***Role*** has the ***name*** attribute with value 'Buyer'. Because we are here considering only ***BinaryCollaborations*** in ebBPSS terminology (or their equivalent in other flow languages), we are only interested in those ***CollaborationRole*** node sets within our collaborator's *CPP* that have a ***Role*** value equal to 'Seller.' So we assume we have narrowed our focus to ***CollaborationRole*** node sets in Company

A's *CPP* with ***Role/@name=***'Buyer' and in Company B's ***CollaborationRole*** node sets with ***Role/@name=***'Seller'.

For more general collaborations, such as in the ***MultiPartyCollaboration***s of ebBPSS, we would need to know the list of roles available within the process, and keep track of that for each of the ***CollaborationRoles***, the ***Role*** values chosen correspond correctly for the participants. We do not here discuss the matching/filtering task for collaborations involving more than two roles, as multiparty *CPA*s are not within scope for version 2.0 of this specification.

E.5 Implementation Matching Tasks

After filtering the CollaborationRoles with the desired ProcessSpecification, we should find one CollaborationRole in our own CPP where we play the Buyer role, and one CollaborationRole in our intended collaborator Company B's CPP where it plays the Seller role.

Our next task is to locate the specific candidate *bindings* relevant to *CPA* formation. There are bindings for Service and Actions. For initial simplicity, we consider detailed matching tasks as they arise for a standard collaboration case involving a *Request* action, followed by a *Response* action. For version 2.0 of this specification, most matching tasks will involve matching of referenced components of the *CPP*'s ***ThisPartyActionBinding*** elements under ***CollaborationRole/ServiceBinding/CanSend/*** and under ***CollaborationRole/ServiceBinding/CanReceive.***

E.5.1 Action Correspondence and Selecting Correlative PackageIds, and ChannelIds

In *CPP*s, under each of the elements, ***CollaborationRole/ServiceBinding/CanSend*** and ***CollaborationRole/ServiceBinding/CanReceive,*** are lists of ***ThisPartyActionBinding***s. For *Request-Response* collaboration patterns, we are interested in matches:

1. in the bindings of the Requesting side's ***CanSend/ThisPartyActionBinding*** with the Responding side's ***CanReceive/ThisPartyActionBinding*** for the request action**,** and
2. in the bindings of the Responding side's ***CanSend/ThisPartyActionBinding*** with the Requesting side's ***CanReceive/ThisPartyActionBinding*** for the response action**.**

These correlative bindings give us references to detailed components that need to match for a fully interoperable agreement. Case 1 pertains to the *Reques*t. Case 2 pertains to the *Response.*

For example, for Company A, we find under ***CanSend***:

```
<tp:ThisPartyActionBinding tp:action="Purchase Order Request
Action" tp:packageId="CompanyA_RequestPackage">
      <tp:BusinessTransactionCharacteristics ... />
     <tp:ActionContext tp:binaryCollaboration="Request Purchase
     Order" tp:businessTransactionActivity="Request Purchase
     Order" tp:requestOrResponseAction="Purchase Order Request
     Action"/>
```

```
      <tp:ChannelId>asyncChannelA1</tp:ChannelId>
</tp:ThisPartyActionBinding>
```

Correlative to this, for Company B, we find under ***CanReceive***:

```
<tp:ThisPartyActionBinding tp:action="Purchase Order Request
Action" tp:packageId="CompanyB_RequestPackage">
       <tp:BusinessTransactionCharacteristics ... />
      <tp:ActionContext tp:binaryCollaboration="Request Purchase Order"
      tp:businessTransactionActivity="Request Purchase Order"
      tp:requestOrResponseAction="Purchase Order Request Action"/>
      <tp:ChannelId>asyncChannelB1</tp:ChannelId>
</tp:ThisPartyActionBinding>
```

The correlation of elements can normally (when we are dealing with BPSS ***Binary Collaboration***s or their equivalents in other representations) be based on equality of the ***action*** (or ***requestOrResponseAction***) values. More detailed correlation of elements can make use of more detailed testing and comparisons of the values in the ***ActionContext*** child elements of the relevant ***CanSend*** and ***CanReceive*** pairs.

In the preceding, we have illustrated the matching of ***CanSend*** and ***CanReceive*** for asynchronous bindings. All ***CanSend*** bindings that are siblings under a ***ServiceBinding*** element are asynchronous and make of use separate TCP connections that the ***CanSend*** side initiates on a listening TCP port. In order to represent binding details for synchronous sending, the convention is adopted whereby the ***CanReceive***element for a Sender is placed under its ***CanSend*** element. This is illustrated by:

```
<tp:CanSend>
   <tp:ThisPartyActionBinding
    tp:id="companyA_ABID6"
    tp:action="Purchase Order Request Action"
    tp:packageId="CompanyA_RequestPackage">
  <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"/>
    <tp:ChannelId>syncChannelA1</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    <tp:CanReceive>
```

```
6551          <tp:ThisPartyActionBinding
6552            tp:id="companyA_ABID7"
6553            tp:action="Purchase Order Confirmation Action"
6554            tp:packageId="CompanyA_SyncReplyPackage">
6555           <tp:BusinessTransactionCharacteristics
6556            tp:isNonRepudiationRequired="true"
6557            tp:isNonRepudiationReceiptRequired="true"
6558            tp:isConfidential="transient"
6559            tp:isAuthenticated="persistent"
6560            tp:isTamperProof="persistent"
6561            tp:isAuthorizationRequired="true"
6562            tp:timeToAcknowledgeReceipt="PT2H"
6563            tp:timeToPerform="P1D"/>
6564           <tp:ActionContext
6565             tp:binaryCollaboration="Request Purchase Order"
6566             tp:businessTransactionActivity="Request Purchase
6567  Order"
6568             tp:requestOrResponseAction="Purchase Order
6569  Confirmation Action"/>
6570          <tp:ChannelId>syncChannelA1</tp:ChannelId>
6571          </tp:ThisPartyActionBinding>
6572      </tp:CanReceive>
6573      <tp:CanReceive>
6574         <tp:ThisPartyActionBinding
6575          tp:id="companyA_ABID8"
6576          tp:action="Exception"
6577          tp:packageId="CompanyA_ExceptionPackage">
6578        <tp:BusinessTransactionCharacteristics
6579          tp:isNonRepudiationRequired="true"
6580          tp:isNonRepudiationReceiptRequired="true"
6581          tp:isConfidential="transient"
6582          tp:isAuthenticated="persistent"
6583          tp:isTamperProof="persistent"
6584          tp:isAuthorizationRequired="true"
6585          tp:timeToAcknowledgeReceipt="PT2H"
6586          tp:timeToPerform="P1D"/>
6587       <tp:ChannelId>syncChannelA1</tp:ChannelId>
6588       </tp:ThisPartyActionBinding>
6589       </tp:CanReceive>
6590    </tp:CanSend>
6591
```

6592 This subordination will also carry over to the synchronous receiving side, in which any of its
6593 ***CanSend*** elements are under the ***CanReceive*** element used to represent the initial receiving of a
6594 request. An illustration from Company B's synchronous binding is:
6595

```
6596  <tp:CanReceive>
6597    <tp:ThisPartyActionBinding
6598    tp:id="companyB_ABID8"
6599    tp:action="Purchase Order Request Action"
6600    tp:packageId="CompanyB_SyncReplyPackage">
6601    <tp:BusinessTransactionCharacteristics
```

```
6602        tp:isNonRepudiationRequired="true"
6603        tp:isNonRepudiationReceiptRequired="true"
6604   tp:isConfidential="transient"
6605        tp:isAuthenticated="persistent"
6606   tp:isTamperProof="persistent"
6607        tp:isAuthorizationRequired="true"
6608   tp:timeToAcknowledgeReceipt="PT5M"
6609        tp:timeToPerform="PT5M"/>
6610     <tp:ActionContext
6611        tp:binaryCollaboration="Request Purchase Order"
6612        tp:businessTransactionActivity="Request Purchase Order"
6613        tp:requestOrResponseAction="Purchase Order Request
6614   Action"/>
6615       <tp:ChannelId>syncChannelB1</tp:ChannelId>
6616       </tp:ThisPartyActionBinding>
6617       <tp:CanSend>
6618         <tp:ThisPartyActionBinding
6619          tp:id="companyB_ABID6"
6620          tp:action="Purchase Order Confirmation Action"
6621          tp:packageId="CompanyB_ResponsePackage">
6622          <tp:BusinessTransactionCharacteristics
6623             tp:isNonRepudiationRequired="true"
6624             tp:isNonRepudiationReceiptRequired="true"
6625                       tp:isConfidential="transient"
6626             tp:isAuthenticated="persistent"
6627             tp:isTamperProof="persistent"
6628             tp:isAuthorizationRequired="true"
6629             tp:timeToAcknowledgeReceipt="PT5M"
6630             tp:timeToPerform="PT5M"/>
6631         <tp:ActionContext
6632           tp:binaryCollaboration="Request Purchase Order"
6633           tp:businessTransactionActivity="Request Purchase Order"
6634           tp:requestOrResponseAction="Purchase Order Confirmation
6635   Action"/>
6636        <tp:ChannelId>syncChannelB1</tp:ChannelId>
6637        </tp:ThisPartyActionBinding>
6638       </tp:CanSend>
6639      <tp:CanSend>
6640        <tp:ThisPartyActionBinding
6641         tp:id="companyB_ABID7"
6642         tp:action="Exception"
6643         tp:packageId="CompanyB_ExceptionPackage">
6644       <tp:BusinessTransactionCharacteristics
6645         tp:isNonRepudiationRequired="true"
6646         tp:isNonRepudiationReceiptRequired="true"
6647         tp:isConfidential="transient"
```

```
6648          tp:isAuthenticated="persistent"
6649          tp:isTamperProof="persistent"
6650          tp:isAuthorizationRequired="true"
6651          tp:timeToAcknowledgeReceipt="PT5M"
6652          tp:timeToPerform="PT5M"/>
6653        <tp:ChannelId>syncChannelB1</tp:ChannelId>
6654         </tp:ThisPartyActionBinding>
6655        </tp:CanSend>
6656    </tp:CanReceive>
6657
```

E.5.2 Matching and Checking DeliveryChannel Details

Until now, most of the matching work has been undertaken to find pairs of correlative action binding, and so the matching has functioned as a filtering mechanism. Once in possession of pairs of correlative action bindings, however, the work of checking for matches across the various dimensions of operation—transport, transport security, PKI compatibility for various tasks, agreement about messaging characteristics (reliable messaging, digital enveloping, signed acknowledgments (minimal non-repudiation of receipt), non-repudiation of origin, packaging details, and more begins.

Once in possession of the action bindings, IDREFs provide references to the underlying components for comparison. For example, when comparing packaging details, the *Request* IDREFS are found at ***CanSend/ThisPartyActionBinding/@packageId*** and within the other *CPP* at ***CanReceive/ThisPartyActionBinding@packageId.*** For Company A's *Request* "Purchase Order Request Action," the packaging IDREF is found in:

```
tp:packageId="CompanyA_RequestPackage"
```

and this IDREF value refers to:

```
<tp:Packaging tp:id="CompanyA_RequestPackage">
     <tp:ProcessingCapabilities tp:parse="true"
tp:generate="true"/>
     <tp:CompositeList>
          <tp:Composite tp:id="CompanyA_RequestMsg"
     tp:mimetype="multipart/related"
     tp:mimeparameters="type=text/xml;">
     <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
     <tp:Constituent tp:idref="CompanyA_Request"/>
     </tp:Composite>
     </tp:CompositeList>
</tp:Packaging>
```

For Company A's *Request* "Purchase Order Request Action", the delivery channel IDREF is found in:

```
<tp:ChannelId>asyncChannelA1</tp:ChannelId>
```

and this IDREF value refers to the element with this ID, namely:

```
<tp:DeliveryChannel tp:channelId="asyncChannelA1"
tp:transportId="transportA1" tp:docExchangeId="docExchangeA1">
<tp:MessagingCharacteristics
     tp:syncReplyMode="none"
     tp:ackRequested="always"
     tp:ackSignatureRequested="always"
     tp:duplicateElimination="always"/>
</tp:DeliveryChannel>
```

Two remaining crucial references for understanding the binding, are found in attributes of the ***DeliveryChannel,*** namely: ***DeliveryChannel/@transportId*** and in the attribute ***DeliveryChannel/@docExchangeId.***

For Company A, for example, we find ***transportId***="transportA1" and ***docExchangeId***="docExchangeA1" are the IDREFs for the continuing binding information with the ***DeliveryChannel, "***asyncChannelA1". Resolving these references, we obtain:

```
<tp:Transport tp:transportId="transportA1">
     <tp:TransportSender>
     <tp:TransportProtocol
     tp:version="1.1">HTTP</tp:TransportProtocol>
     <tp:TransportClientSecurity>
          <tp:TransportSecurityProtocol
     tp:version="3.0">SSL</tp:TransportSecurityProtocol>
     <ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
          <tp:ServerSecurityDetailsRef
     tp:securityId="CompanyA_TransportSecurity"/>
     </tp:TransportClientSecurity>
     </tp:TransportSender>
     <tp:TransportReceiver>
     <tp:TransportProtocol
     tp:version="1.1">HTTP</tp:TransportProtocol>
     <tp:Endpoint
     tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/asyn
     c" 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:TransportServerSecurity>
     </tp:TransportReceiver>
```

```
</tp:Transport>
```

for ***transportID*** "transportA1" and

```
<tp:DocExchange tp:docExchangeId="docExchangeA1">
     <tp:ebXMLSenderBinding tp:version="2.0">
     <tp:ReliableMessaging>
     <tp:Retries>3</tp:Retries>
     <tp:RetryInterval>PT2H</tp:RetryInterval>
     <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemant
ics>
     </tp:ReliableMessaging>
     <tp:PersistDuration>P1D</tp:PersistDuration>
     <tp:SenderNonRepudiation>
     <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsi
g#
      </tp:NonRepudiationProtocol>
     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1
     </tp:HashFunction>
     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#ds
a-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:ebXMLSenderBinding>
     <tp:ebXMLReceiverBinding tp:version="2.0">
     <tp:ReliableMessaging>
     <tp:Retries>3</tp:Retries>
     <tp:RetryInterval>PT2H</tp:RetryInterval>
     <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemant
ics>
     </tp:ReliableMessaging>
     <tp:PersistDuration>P1D</tp:PersistDuration>
     <tp:ReceiverNonRepudiation>
     <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsi
g#
     </tp:NonRepudiationProtocol>
     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1
```

```
6786        </tp:HashFunction>
6787        <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#ds
6788    a-sha1
6789        </tp:SignatureAlgorithm>
6790        <tp:SigningSecurityDetailsRef
6791        tp:securityId="CompanyA_MessageSecurity"/>
6792        </tp:ReceiverNonRepudiation>
6793        <tp:ReceiverDigitalEnvelope>
6794        <tp:DigitalEnvelopeProtocol
6795        tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
6796        <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
6797        <tp:EncryptionCertificateRef
6798        tp:certId="CompanyA_EncryptionCert"/>
6799        </tp:ReceiverDigitalEnvelope>
6800        </tp:ebXMLReceiverBinding>
6801    </tp:DocExchange>
```

6802

6803 for the ***docExchangeId***, docExchangeA1.

6804

6805 There are, of course, other references, such as those to security-related capabilities, that will be
6806 important to resolve when checking detailed matching properties, but the four IDREFs (two for
6807 the sender and two for the receiver) that have just been introduced are critical to the remainder of
6808 the match tests that will lead to the formation of draft *CPA*s**.** We will assume at this point that the
6809 reader can resolve IDREFs using the example *CPP*s and *CPA*s for Company A and B in the
6810 appendices, and will not exhibit them in the text in order to save space.

6811

6812 We next turn to a more in-depth treatment of the tests that are involved in finding the elements
6813 for a draft *CPA***.**

6814

6815 The detailed tasks to be discussed in greater depth are:

6816

6817 1. Matching *Channel* ***MessagingCharacteristics***
6818 2. Checking ***BusinessTransactionCharacteristics*** coherence with *Channel* details
6819 3. Matching ***Packaging***
6820 4. Matching ***Transport*** and ***Transport[Receiver|Sender]Security***
6821 5. Matching and Checking ***DocExchange*** subtrees.

6822

6823 Because agreement about ***Transport*** is quite fundamental, we shall consider it first.
6824 Computational processes are likely to first find pairs that match on ***Transport*** details, and will
6825 ignore pairs failing to have matches at this level.

6826

6827 **E.5.2.1 Matching Transport**

6828 Matching ***Transport*** first involves matching the ***Transport/TransportSender/TransportProtocol***
6829 capabilities of the requester with the ***Transport/TransportReceiver/TransportProtocol***
6830 capabilities found under the collaborator receiving the *request*. Several such matches can exist,

and any of these matches can be used in forming a draft, provided other aspects match up satisfactorily. Each *CPP* is assumed to have listed its preferred transport protocols first (as determined by the listing of the Bindings that reference the ***Transport*** element, but different outcomes can result depending on which *CPP* is used first for searching for matches. In general, resolution of preference differences is left to a distinct phase of *CPA negotiation,* following proposal of a draft *CPA.* Negotiation can be performed by explicit actions of users, but is expected to become increasingly automated.

Matching transport secondly involves matching the ***TransportSender/TransportProtocol*** capabilities of the responding collaborator with its ***TransportReceiver/TransportProtocol*** capabilities found under the collaborator receiving the response, which is typically the collaborator that has sent a request. Several such matches can exist, and any of these matches can be used in forming a draft. In one case, however, there may be no need for the second match on ***TransportProtocol.*** If we are using HTTP or some other protocol supporting synchronous replies, and the ***DeliveryChannel*** has a ***MessagingCharacteristics*** child that has its ***syncReplyMode*** attribute with a value of "signalsAndResponse," then everything comes back synchronously, and there is no need to match on ***TransportProtocol*** for the *Response* ***DeliveryChannel.***

If ***TransportSecurity*** is present, then there can be additional checks. First, ***TransportSender/TransportClientSecurity/TransportSecurityProtocol*** should be compatible with ***TransportReceiver/TransportServerSecurity/TransportSecurityProtocol.*** Second, if either the ***TransportSender/TransportClientSecurity/ClientSecurityDetailsRef*** or ***TransportSender/TransportClientSecurity/ServerSecurityDetailsRef*** elements are present, and the IDREF references an element containing some ***AnchorCertificateRef,*** then an opportunity exists to check suitability of one *Party's* PKI trust of the certificates used in the ***TransportSecurityProtocol.*** For example, by resolving the IDREF value in ***TransportSender/TransportClientSecurity/ClientCertificateRef/@certId,*** we can obtain the proposed client certificate to use for client-side authentication. By resolving the IDREFs from the ***AnchorCertificateRef,*** we become able to determine whether the proposed client certificate will "chain to a trusted root" on the server side's PKI. Similar remarks apply to checks on the validity of a server certificate found by resolving ***TransportReceiver/TransportServerSecurity/ServerCertificateRef*** . This server certificate can be checked against the CA trust anchors that are found by resolving ***TransportSender/TransportClientSecurity/ServerSecurityDetailsRef/@securityId,*** and finding CA certificates (or CA certificate chains) in the ***KeyInfo*** elements under the ***Certificate*** element obtained by resolving the IDREF found in ***AnchorCertificateRef@certId.***

When matches exist for the correlative ***Transport*** components, we then have discovered an interoperable solution at the transport level. If not, no *CPA* will be available, and a gap has been identified that will need to be remedied by whatever exception handling procedures are in place. Let us next consider other capabilities that need to match for "thicker" interoperable solutions.

E.5.2.2 Checking BusinessTransactionCharacteristics and DeliveryChannel MessagingCharacteristics

Under each of the correlative action bindings, there is a child element of ***DeliveryChannel, MessagingCharacteristics*** that has several attributes important in *CPA* formation tasks. The attributes having wider implications are ***syncReplyMode, ackRequested,*** and ***ackSignatureRequested;*** for the ***duplicateElimination*** and ***actor*** attributes, compatibility exists when the attributes that are found under the ***CanSend*** and ***CanReceive DeliveryChannels*** have the same values. As the element's name implies, all of these ***DeliveryChannel*** features pertain to the messaging layer.

In addition, ***BusinessTransactionCharacteristics,*** found under ***ThisPartyActionBinding***, contains attributes reflecting a variety of features pertaining to desired security and business transaction properties that are to be implemented by the agreed upon ***DeliveryChannels.*** These properties may have implications on what capabilities are needed within more detailed components of the ***DeliveryChannel*** elements, such as in the ***Packaging*** element. When using a *BPSS* process specification, these properties may be specified within the BusinessTransaction. The properties of the ***BusinessTransactionCharacteristics*** element are, however, the ones that will be operative in the implementation of the ***BusinessTransaction***, and may override the specified values found in the *BPSS* Process specification. Because the properties are diverse, the details that implement the properties can be spread over other elements referenced within the ***DeliveryChannel*** elements.

These attributes apply to either a *Request* or *Response* delivery channel, but can impact either the *Sender* or *Receiver* (or both) in a channel. In addition, the attributes governing acknowledgments, for example, qualify the interrelation of ***DeliveryChannel*** elements by specifying behavior that is to occur that qualifies the contents of a return message.

The most basic test for compatibility for any of the attributes in either ***MessagingCharacteristics*** or ***BusinessTransactionCharacteristics*** is that the attributes are equal in the sending party's ***DeliveryChannel*** referenced by ***CanSend/ThisPartyActionBinding/ChannelId*** and in the receiving party's ***DeliveryChannel*** referenced by ***CanReceive/ThisPartyActionBinding/ChannelId.*** If they are unequal, and all Bindings have been examined on both sides, a draft *CPA* will represent a compromise to some common set with respect to the functionality represented by the attributes.

In the following discussions, we will consider many of the attributes in the two *Characteristics* elements, and relate them to additional underlying implementational details, one of which is ***Packaging***.

From a high level, basic agreement in *packaging* is a matter of compatibility of the generated *packaging* on the sending side with the parsed packaging on the receiving side. The basic packaging check is, therefore, checking packaging compatibility under the ***CanSend*** element of a sender action with the packaging under the ***CanReceive*** element of that same action under the receiver side.

For efficiency, representation of capabilities of parsing/handling packaging can make use of both wildcards and repetition, and as needed these capabilities can also express open data formatting used on the generating side. For example, consider the ***SimplePart:***

```
<tp:SimplePart tp:id="IWild" tp:mimetype="*/*"/>
```

By wildcarding ***mimetype*** values, we represent our capability of accepting any data, and would match any specific MIME type. Also, consider a ***Constituent*** appearing within a ***Composite***:

```
 <tp:Constituent tp:idref="MsgHdr"/>
<tp:Constituent minOccurs="0"
maxOccurs="10" tp:idref="IWild"/>
```

This notation serves to capture the capability of handling any number of arbitrary MIME bodyparts within the ***Composite*** being defined. A Packaging capability such as this would obviously match numerous more specific generated *Packaging* schemes, as well as matching literally with a scheme of the same generality.

Certain more complex checks are needed for more complicated packaging options pertaining to syncReplyMode. These are discussed in the following.

syncReplyMode

The ***syncReplyMode*** has a value other than “none” to indicate what parts of a message should be returned in the *Reply* of a transport capable of synchronous operation, such as HTTP. (We here use “synchronous” to mean “on the same TCP connection,” which is one use of this term. We do not specify any waiting, notification, or blocking behavior on processes or threads that are involved, though presumably there is some computational activity that maintains the connection state and is above the TCP and socket layers.)

The possible implementations pertaining to various values of the ***syncReplyMode****s* are numerous, but we will try to indicate at least the main factors that are involved.

As will be seen, the ***Packaging*** element is important in specifying implementation details and compatibilities. But, because business level signals may be involved, other action bindings may need examination in addition to the already selected bindings for the *Request* and *Response.* Also, the values of ***TransportReceiver/Endpoint/@type*** might need checking when producing draft *CPA****s.***

- Let us first begin with the cases in which *Responses, Message Service Handler Signals* and *Business Signals* return in some combination of a synchronous reply and other asynchronous message(s). These various combinations will be discussed for the ***syncReplyMode*** values: "mshSignalsOnly,” "signalsOnly,” “responseOnly," and "signalsAndResponse."
-
- By convention, synchronous replies are represented by subordinating ***CanSend*** or

CanReceive elements under the ***CanReceive*** or ***CanSend*** elements that represent the initial *Request* binding capabilities. For representing asynchronous Requests, Replies, or Signals, the ***CanSend*** or ***CanReceive*** elements are all siblings and directly subordinate to the ***ServiceBinding.*** Therefore, both asynchronous and synchronous capabilities can be grouped under a ***ServiceBinding*** in a *CPP*, and can still be unambiguously distinguished. In principle, increasing subordination (nesting) can indicate patterns of dialog more elaborate than *Request* and *Response.* Few use cases for this functionality are common at the time of this writing.

mshSignalsOnly

The Request sender's ***DeliveryChannel*** (referenced by ***CanSend/ThisPartyActionBinding/ChannelId***) and the Request receiver's ***DeliveryChannel*** (referenced by ***CanReceive/ThisPartyActionBinding/ChannelId***) both should have ***MessagingCharacteristics/@syncReplyMode*** value of ***mshSignalsOnly***.

While a Party can explicitly identify a ***DeliveryChannel*** for the SOAP envelope with subordinate ***CanSend*** and ***CanReceive*** elements, and with them specialized bindings, these are typically omitted for ebXML Messaging software. It is presumed that each side can process a synchronous reply constructed in accordance with ebXML Messaging. The ***DeliveryChannel*** representation mechanism here serves as a placeholder for capturing other Messaging Signal protocols that might emerge.

Currently acknowledgments and signed acknowledgments, along with errors, are the primary MSH signals that are included in the SOAP envelope. If Company A set ***syncReplyMode*** to ***mshSignalsOnly,*** then Company B's correlative ***CanReceive/ThisPartyActionBinding/@packageId*** should contain a nested ***CanSend/ThisPartyActionBinding/@packageId*** for a message without any business payload or signals. In addition, the ***CanSend/ThisPartyActionBinding/@packageId*** of Company B's *Response* should resolve to packaging format capable of returning the *Response (* and possibly other constituents) asynchronously. The compatibility of the ***DeliveryChannel*** elements can be checked, as can the capability of Company A to receive that Response payload, the Signal payload(s), or Responses bundled with signals as specified by the packaging formats that are referenced through the relevant ***ThisPartyActionBindings*** element's ***packageId*** attribute values.

signalsOnly

The Request sender's ***DeliveryChannel*** (referenced by its ***CanSend/ThisPartyActionBinding/ChannelId***) and the Request receiver's ***DeliveryChannel*** (referenced by its ***CanReceive/ThisPartyActionBinding/ChannelId***) both should have ***MessagingCharacteristics/@syncReplyMode*** value of ***signalsOnly.***

If Company A sets ***syncReplyMode*** to ***signalsOnly,*** then under Company B's correlative ***CanReceive*** element, there should be a nested ***CanSend/ThisPartyActionBinding*** whose ***packageId*** attribute's value resolves to a packaging format appropriate for Signals. For the ***CanSend/ThisPartyActionBinding/@packageId*** associated with Company B's business level *Response,* the attribute IDREF value should resolve to a packaging format capable of returning

payloads and that omits business signals. This ***CanSend*** element will be a direct child of ***ServiceBinding,*** a placement representing its asynchronous character. The original requesting party will need to have a ***CanReceive/ThisPartyActionBinding*** that is compatible with the responding party, and that is a direct child of its ***ServiceBinding*** element.

Using subordinate ***CanSend*** and subordinate ***CanReceive*** elements can be useful if the ***DeliveryChannel*** details for Exception signals differ from those specified for Request and Response. Signal bindings, for example, may differ by omitting ***ackRequested,*** or possibly one of the security features (digital enveloping or non-repudiation of receipt) that are used for Requests or Responses. Just as with other tests on Requests and Responses, there can be checks for compatibility in ***Packaging, DocExchange, MessagingCharacteristics,*** or ***BusinessTransactionCharacteristics*** referred to in the correlative subordinate ***CanSend*** and ***CanReceive DeliveryChannels.***

responseOnly

The Request sender's ***DeliveryChannel*** (referenced by ***CanSend/ThisPartyActionBinding/ChannelId***) and the Request receiver's ***DeliveryChannel*** (referenced by ***CanReceive/ThisPartyActionBinding/ChannelId***) both should have ***MessagingCharacteristics/@syncReplyMode*** value of ***responseOnly.***

If Company A sets ***syncReplyMode*** to ***responseOnly,*** the ***CanSend/ThisPartyActionBinding/@packageId*** of Company B's response should resolve to a packaging format capable of returning payloads, but omitting business signals. The ***CanSend/ThisPartyActionBinding*** element will be included as a child of the ***CanReceive*** element so the responder can indicate that it is a synchronous response.

There should be an independent way to return business level error signals. So, there should be a ***ThisPartyActionBinding*** for any Signal payload announced, and these bindings should be at the direct child of ***ServiceBinding*** level to represent their asynchronous flavor.

It is not too likely that *ReceiptAcknowledgment* and similar signals will be used when a response is returned synchronously. The motivation for using these signals is indicating positive forward progress, and this motivation will be undermined when a Response is returned directly.

For the ***responseOnly*** case, including subordinate ***CanSend/ThisPartyActionBinding*** and ***CanReceive/ThisPartyActionBinding,*** means that there can be checks for compatibility in ***Packaging, DocExchange, MessagingCharacteristics,*** or ***BusinessTransactionCharacteristics.*** The ***syncReplyMode*** and ***ackRequested*** attributes here should be carefully considered because a ***mshSignalsOnly*** value here would mean that another round of synchronous messaging will need to occur on the same connection. Incidentally, for ***Transport*** elements referenced under subordinate bindings, there need not be any ***Endpoint*** elements. If there are ***Endpoint*** elements, they may be ignored.

signalsAndResponse

The Request sender's ***DeliveryChannel*** (referenced by ***CanSend/ThisPartyActionBinding/ChannelId***) and the Request receiver's ***DeliveryChannel*** (referenced by ***CanReceive/ThisPartyActionBinding/ChannelId***) both should have ***MessagingCharacteristics/@syncReplyMode*** value of ***signalsAndResponse.***

If Company A sets ***syncReplyMode*** to ***signalsAndResponse,*** the ***CanSend/ThisPartyActionBinding*** of Company B's response should be subordinate to Company B's ***CanReceive*** element. The packaging format that is referenced should be capable of returning payloads and signals bundled together. If no asynchronous bindings exist for error signals, this will be the only defined ***DeliveryChannel*** agreed to for all aspects of message exchange for the business transaction. However, it is likely that an asynchronous binding would normally be provided to send Exception signals.

ackRequested and ackSignatureRequested

Checks on the ***ackRequested*** and ***ackSignatureRequested*** attributes within correlative ***DeliveryChannels*** (that is, correlative because referenced under one action's ***CanSend*** and ***CanReceive*** elements) are primarily to see that the values of the corresponding attributes are the same.

However, there are some interactions of these attributes with other information items that need to be mentioned.

The principal use of the ***ackRequested*** attribute is within reliable messaging configurations. If reliable messaging is to be configured, then checks on agreement in the correlative ***ReliableMessaging*** elements as found under ***DocExchange/ebXMLSenderBinding*** and ***DocExchange/ebXMLReceiverBinding*** are in order. Also, the value of the ***duplicateElimination*** attribute of ***MessagingCharacteristics*** should be checked for agreement. Draft *CPAs* may be formed by deliberately aligning values that are not equal along some of these dimensions. Downgrading may provide draft *CPA*s most likely to gain acceptance; so, for example, if ***duplicateElimination*** is false on the receiving side, aligning it to false on the sending side is most likely to produce a draft that succeeds.

The additional function of ***ackSignatureRequested*** is that it provides a "thin" implementation for *non-repudiation of receipt.* The basic check is for equality of attribute value, but additional constraints may need test and alignment. If no signal capable of implementing *non-repudiation of receipt* is found under the ***ServiceBinding,*** then having an "always" value for ***ackSignatureRequested*** suggests aligning the ***BusinessTransactionCharacteristics*** attributes, ***isNonRepudiationReceiptRequired,*** to be true. However, if this is done, care should be taken to check that the ***BusinessTransactionCharacteristics*** attribute ***isIntelligibleCheckRequired*** is false. This is because the messaging implementation only deals with receipt in the sense of having received a byte stream off the wire (and persisting it so that it is available for further processing). It is not safe to presume that any syntactical or semantic checks on the data were performed.

E.5.2.3 DocExchange Checks for BusinessTransactionCharacteristics

When using *CPPs* and *CPAs* with ebXML Messaging, which is the most likely early deployment situation, there exists an opportunity to check agreement on ***BusinessTransactionCharacteristics*** attributes:

The following three attributes need to have equal values in the bindings for a Request or for a Response. No further discussion will be provided in this appendix on these "deadlines," except to say that a sophisticated proposed *CPA* generation tool might check on the coherence of the values chosen here with values for reliable messaging parameters, existence of compatible ReceiptAcknowledgment or AcceptanceAcknowledgment bindings, and consistency with syncReplyMode internal configuration.

```
<attribute name="timeToAcknowledgeReceipt" type="duration"/>
<attribute name="timeToAcknowledgeAcceptance" type="duration"/>
<attribute name="timeToPerform" type="duration"/>
```

The remaining attributes involve a number of security related issues and will be the focus of the remaining discussion of ***BusinessTransactionCharacteristics*** attributes:

```
<attribute name="isNonRepudiationRequired" type="boolean"/>
<attribute name="isNonRepudiationReceiptRequired"
type="boolean"/>
<attribute name="isIntelligibleCheckRequired" type="boolean"/>
<attribute name="isAuthenticated"
type="tns:persistenceLevel.type"/>
<attribute name="isTamperProof"
type="tns:persistenceLevel.type"/>
<attribute name="isAuthorizationRequired" type="boolean"/>
<attribute name="isConfidential"
type="tns:persistenceLevel.type"/>
```

Here, the basic test is that for correlative ***DeliveryChannels,*** the corresponding attributes have the same values. Again there are some interaction aspects with parts of the ***DeliveryChannel*** that motivate making some additional checks.

Previously, when discussing the ***MessagingCharacteristics*** attribute ***ackSignatureRequested,*** it was pointed out that the messaging implementation provides thin support for holding ***isNonRepudiationReceiptRequired*** true provided that the attribute ***isIntelligibleCheckRequired*** is false. When both are true, then there should exist a business signal with compatible ***Packaging*** and ***DeliveryChannel*** values. If the signal has been independently described within asynchronous ***CanSend*** and ***CanReceive*** elements, knowing the signal name (such as, "ReceiptAcknowlegment") may support a relatively simple search and test. However, if synchronous transports are involved, some filters using ***syncReplyModes*** may be needed to discover an underlying support for a "thick" implementation of *non-repudiation of receipt*.

When non-repudiation of receipt is implemented by a business signal, then checks on signing certificate validity can involve the ***CollaborationRole/ApplicationCertificateRef*** and the ***CollaborationRole/ApplicationSecurityDetailsRef***, that provides a reference to the ***SecurityDetails*** element containing the list of ***TrustAnchors***. The certificate from the side

signing the ReceiptAcknowledgment would be checked against the certificates referred to by the ***AnchorCertificateRef*** under ***TrustAnchors.***

The business signal will sometimes be conveyed as part of a message. It remains true that the message itself will still be sent through a MSH, and that the MSH can also sign the message using the certificate found by resolving the IDREF found at ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId.***

If a particular software component implements both MSH functionality and business level security functionality, it is possible that the same certificate may be pointed to by ***ApplicationCertificateRef*** and ***SigningCertificateRef/@certId.*** In other words, the distinction between MSH level signing and application level signing is a logical one, and may not correspond with software component boundaries. Because the MSH signature is over the message, the message signature may be over an application level signature. While this may be redundant for some system configurations, protocols may require both signatures to exist over the different regions.

Failure to validate a certificate may not prevent formation of a draft *CPA*. First, the sender's signing certificate can be a self-signed certificate. If so, a reference to this self-signed certificate may be added to the receiver's ***TrustAnchors/AnchorCertificateRef*** list. This proposal amounts to proposing to agree to a direct trust model, rather than a hierarchical model involving certificate authorities. Second, a proposal to add a trusted root may be made, again by appropriate revision of the ***TrustAnchors.***

When non-repudiation of receipt is implemented by the Messaging layer, the checks on PKI make use of elements under ***DocExchange.***

isNonRepudiationRequired
isAuthenticated
isAuthorizationRequired
isTamperProof

The ideas of authentication, authorization, nonrepudiation and being "tamper proof" may be very distinct as business level concepts, yet the implementation of these factors tend to use very similar technologies. Actually, prevention of tampering is not literally implemented. Instead, means are provided for detecting that tampering (or some accidental garbling) has occurred. Likewise, implementations of authorization usually are provided by implementations of access control (permitting or prohibiting a user in a role making use of a resource) and presentation of a token or credential to gain access, which may involve authentication as an initial step! Nonrepudiation may build on all the previous functions, plus retaining information for supplying presumptive evidence of origination at some later time.

When checking whether ***isNonRepudiationRequired*** can be set to True for both *Parties*, check whether the signing certificate will be counted as valid at the receiver.

The IDREF reference to the signing certificate is found in ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId.*** The referenced certificate should be checked for validity with respect to the trust anchors obtained from ***TrustAnchors/AnchorCertificateRef*** elements under the ***SecurityDetails*** element referenced by the IDREF at ***DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation/SigningSecurityDetailsRef/@securityId.***

As previously noted, failure to validate a certificate does not prevent constructing a draft CPA. Either self-signed certificates or new trust anchors can be added to align the trust model on one side with the other side's certificate.

In addition to checking the interoperability of the PKI infrastructures, checks on compatibility of values in the other attributes in ***DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation*** and in ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation*** can be made***.*** ***NonRepudiationProtocol, HashFunction,*** and ***SignatureAlgorithm*** values may be compatible even when not equal if knowledge of the protocol requirements allows fallback to a mandatory to implement value. So values here can be found equal, aligned, or negotiated to reach an agreement.

If ***isNonRepudiationRequired*** is True, the ***isAuthenticated*** and ***isTamperProof*** should also be True. This is because in implementing ***isNonRepudiationRequired*** by means of a digital signature, both authentication (with respect to the identity associated with the signing certificate) and tamper detection (with respect to the cryptographic hash of the signature) will be implemented as well. The converses need not be true because authentication and tamper detection might be accomplished without archiving information needed to support claims of nonrepudiation.

isConfidential

The ***isConfidential*** attribute indicates properties variously distributed among levels of the application-to-application sending/receiving stacks.

isConfidential has possible values of "none", "transient", "persistent", and "transient-and-persistent. The "persistent" or "transient-and-persistent" values indicate that some digital enveloping function is present; a "transient" value indicates confidentiality is applied at the transfer layer or below.

ebXML Messaging version 2.0 does not have an "official" implementation for digital envelopes, and refers to the future XML Encryption specification as its intended direction for that function. However, the XML Encryption specification is now a candidate recommendation, and is suitable for preliminary implementation.

Within the *CPA,* the ***DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope*** and ***DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope*** can provide configuration details pertaining to security in accordance with [XMLENC].Use of XML Encryption also will normally show up in the value of ***DigitalEnvelopeProtocol,*** and can also appear within a ***NamespaceSupported*** element within ***Packaging.***

Currently, [ebMS] has only indicated a direction to eventually use XML Encryption, but has not mandated any digital envelope protocol. Digital enveloping may be done at the "application level," and will show up under MIME types within the ***Packaging*** element. PKI matching will make use of certificates supplied in ***ApplicationCertificateRef*** and ***ApplicationSecurityDetailsRef.*** If other protocols are to be used, it would be safest to use extensions to the content model of ***DocExchange,*** such as, ***XXXSenderBinding*** and ***XXXReceiverBinding***, and follow the pattern of the ebXML content models for ***DocExchange.*** Future versions of this specification intend to make these extension semantics easier to use interoperably; currently, the extensions would be a multilateral extension within some trading community.

When checking whether ***isConfidential*** can be set to "persistent" or "transient-and-persistent" for both *Parties*, check whether the key exchange certificate will be counted as valid at the sender. The IDREF reference to the ***SecurityDetails*** element is found in ***DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope/EncryptionSecurityDetailsRef/@securityId.*** The trust anchor certificates obtained from ***TrustAnchors/AnchorCertificateRef*** elements under the ***SecurityDetails*** element will be used to test that the certificate referenced by ***DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@certId*** validates at the sender side.

As previously noted, failure to validate a certificate does not prevent constructing a draft CPA. Either self-signed certificates or new trust anchors can be added to align the trust model on one side with the other side's certificate.

In addition to the PKI related checks and alignments, the elements ***EncryptionAlgorithm*** and ***DigitalEnvelopeProtocol*** should be checked for equality (or compatibility) and, if not compatible or equal, aligned to values that would work for an initial version of a proposed *CPA*. Preferences and alignment of these elements can be achieved in a subsequent Negotiation phase.

Finally, it is possible that one side's DigitalEnvelope will be modeled using either the ***DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope*** and ***DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope,*** while the other side uses only ***Packaging*** to indicate use of, for example, S/MIME Digital Envelopes, because it receives an already enveloped payload from an application. In such a case, the PKI certificate validation check could require checking that a certificate described by ***DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@certId*** validates against the ***TrustAnchors*** found by resolving ***CollaborationRole/ApplicationSecurityDetailsRef.*** This complication arises from the possibility

that digital enveloping functionality can be spread over quite distinct portions of the stack in different software installations.

E.6 CPA Formation: Technical Details

When assembling a draft *CPA* from matching portions of two *CPP*s' ***PartyInfo*** elements, some additional constraints need to be observed.

First, as mentioned in section 9.11.1, software for producing draft CPAs needs to guarantee that ID values in one *CPP* are distinct from ID values in the other *CPP* so that no IDREF references collide when the *CPP*s are merged. The following ID values are potentially subject to collision:

Certificates
SecurityDetails
SimplePart
Packaging
DocExchange
Transport
DeliveryChannel
ThisPartyActionBinding

There are elements and complex type definitions containing IDREFs. Also some elements have attributes with IDREF values. These are:

PartyInfo
ActionBinding.type
ThisPartyActionBinding
OtherPartyActionBinding
OverrideMSHActionBinding
ChannelId
DeliveryChannel
Constituent
CertificateRef.type
AnchorCertificateRef
ApplicationCertificateRef
ClientCertificateRef
ServerCertificateRef
SigningCertificateRef
EncryptionCertificateRef
CertificateRef
SecurityDetailsRef.type

Second, when the ***CanSend*** and ***CanReceive*** binding information has been found to match (equal, correspond with, or be compatible with) the binding information under the other Party's ***CanReceive*** and ***CanSend*** elements, the IDREF references for the ***OtherPartyActionBinding***

7322 are filled out in the CPA.
7323
7324 Third, for CPAs that are signed, the implementer is advised to review section 9.9.1.1 when using
7325 [XMLDSIG] for the signature technique. A proposed CPA need not have a signature.
7326
7327 Fourth, when a *CPA* is composed from two *CPPs*, see section 8.8 in which it stated that all
7328 ***Comment*** elements from both *CPPs* SHALL be included in the *CPA* unless agreed to otherwise.
7329
7330 Fifth, several tests on CPA validity could be conducted on draft CPAs, but these tests are more
7331 critical for a negotiated CPA that is to be deployed and imported into run-time software
7332 components.
7333
7334 1. Expiration: Certificates used in signing a CPA can be checked to verify that they do not expire
7335 before the CPA expires, as given in the ***End*** element.
7336
7337 2. Certificate expiration: If a CPA lifetime exceeds the lifetime of certificates accepted for use in
7338 signing, key exchange or other security functions, then it would be advisable to make ds:KeyInfo
7339 refer to certificates, rather than to include them within the element by value.
7340
7341 3. Process-Specification references can be checked in accordance with the provisions of section
7342 8.4.4 and its subsections.
7343
7344 Finally, a CPA has several elements whose values are not typically derived from either CPPs
7345 (and can need checking when using a CPA template as the basis for a draft CPA.) The ***Status,***
7346 ***Start, End,*** and possibly a ***ConversationConstraints*** element need to be added. The attributes,
7347
7348 ***CollaborationProtocolAgreement/@cpaid,***
7349 ***CollaborationProtocolAgreement/@version,***
7350 ***CollaborationProtocolAgreement/Status@value,***
7351 ***CollaborationProtocolAgreement/ConversationConstrain@invocationLimit,*** and
7352 ***CollaborationProtocolAgreement/ConversationConstraint@concurrentConversations,***
7353
7354 can also be supplied values as needed.

Appendix F Correspondence Between CPA and ebXML Messaging Parameters (Normative)

The following table shows the correspondence between elements used in the ebXML Messaging Service message header and their counterparts in the CPA.

Message Header Element / Attribute	Corresponding CPA Element / Attribute
PartyId element	***PartyId*** element; if multiple ***PartyID*** elements occur under the same ***PartyInfo*** element in the *CPA*, all of them MUST be included in the *Message Header*
Role element	***Role*** element
CPAId element	***cpaid*** attribute in ***CollaborationProtocolAgreement*** element
ConversationId element	No equivalent; SHOULD be generated by software above the Message Service Interface (MSI)
Service element	***Service*** element
Action element	***action*** attribute in ***ThisPartyActionBinding*** element
TimeToLive element	Computed as the sum of ***Timestamp*** (in message header) + ***PersistDuration*** (under ***DocExchange***/***ebXMLReceiverBinding***)
MessageId element	No equivalent; generated by the MSH per message
Timestamp element	No equivalent; generated by the MSH per message
RefToMessageId element	No equivalent; usually passed in by the application where applicable; SHOULD be used for correlating response messages with request messages
SyncReply element	***syncReplyMode*** attribute in ***MessagingCharacteristics*** element; the ***SyncReply*** element is included if and only if the ***syncReplyMode*** attribute is not “none”
DuplicateElimination element	***duplicateElimination*** attribute in ***MessagingCharacteristics*** element; the ***DuplicateElimination*** element is included if the ***duplicateElimination*** attribute under ***MessagingCharacteristics*** is set to “always”, or if it is set to “perMessage” and the application indicates to the MSH that duplicate elimination is desired

Manifest element	***Packaging*** element; each ***Reference*** element under ***Manifest*** SHOULD correspond to a ***SimplePart*** that is referenced from one of the ***CompositeList*** elements under ***Packaging***
xlink:role attribute in ***Reference*** element	***xlink:role*** attribute in ***SimplePart*** element
AckRequested element	***ackRequested*** attribute in ***MessagingCharacteristics*** element; an ***AckRequested*** element is included in the SOAP Header if the ***ackRequested*** attribute is set to "always"; if it is set to "perMessage", input passed to the MSI is to be used to determine if an ***AckRequested*** element needs to be included; likewise, the signed attribute under ***AckRequested*** will be appropriately set based on the ***ackSignatureRequested*** attribute and possibly determined by input passed to the MSI
MessageOrder element	***messageOrderSemantics*** attribute in ***ReliableMessaging*** element; the ***MessageOrder*** element will be present if the ***AckRequested*** element is present, and if the ***messageOrderSemantics*** attibute in the ReliableMessaging element is set to "Guaranteed"
ds:Signature element	***ds:Signature*** will be present in the SOAP Header if the isNonRepudiationRequired attribute in the ***BusinessTransactionCharacterisitcs*** element is set to "true"; the relevant parameters for constructing the signature can be obtained from the ***SenderNonRepudiation*** and ***ReceiverNonRepudiation*** elements

7361
7362 The following table shows the implicit parameters employed by the ebXML Messaging Service
7363 that are not included in the message header and how those parameters can be obtained from the
7364 CPA.
7365

Implicit Messaging Parameters	**Corresponding CPA Element / Attribute**
Retries (not in Message Header) but used to govern Reliable Messaging behavior in sender	***Retries*** element (under ***ReliableMessaging*** element)
RetryInterval (not in Message Header) but used to govern Reliable Messaging behavior in sender	***RetryInterval*** element (under ***ReliableMessaging*** element)
PersistDuration (not in Message Header) but used to govern Reliable Messaging behavior in receiver	***PersistDuration*** element (under ***ebXMLReceiverBinding*** element)

Endpoint (not in Message Header) but used for sending SOAP message	***Endpoint*** element (under ***TransportReceiver***); the type of message being sent MUST be passed in to the MSI; an appropriate endpoint can then be selected from among the ***Endpoints*** included under the ***TransportReceiver*** element
Use ***Service*** & ***Action*** to determine the corresponding ***DeliveryChannel***	***DeliveryChannel***
Use ***ReceiverDigitalEnvelope*** to determine the encryption algorithm and key	***ReceiverDigitalEnvelope***
Use ***SenderNonRepudiation*** to determine signing certificate(s) and ***ReceiverNonRepudiation*** to determine the trust anchors and security policy to apply to the signing certificate	***SenderNonRepudiation*** and ***ReceiverNonRepudiation***
Use ***Packaging*** to determine how payload containers ought to be encapsulated. Also use ***Packaging*** to determine how an individual SimplePart ought to be extracted and validated against its schema	***Packaging***
Use ***TransportClientSecurity*** and ***TransportServerSecurity*** to determine certificates to be used by server and client for authentication purposes	***TransportClientSecurity*** and ***TransportServerSecurity***
Use the ***DeliveryChannel*** identified by ***defaultMshChannelId*** for standalone MSH level messages like Acknowledgment, Error, StatusRequest, StatusResponse, Ping, Pong, unless overridden by ***OverrideMshActionBinding***	***defaultMshChannelId*** attribute in ***PartyInfo*** element, and ***OverrideMshActionBinding***

7366 Appendix G Glossary of Terms

7367

Term	Definition
AGREEMENT	An arrangement between two partners that specifies in advance the conditions under which they will trade (terms of shipment, terms of payment, collaboration protocols, etc.) An agreement does not imply specific economic commitments.
APPLICATION	Software above the level of the MSH that implements a Service by processing one or more of the Messages in the Document Exchanges associated with the Service.
AUTHORIZATION	A right or a permission that is granted to a system entity to access a system resource.
BUSINESS ACTIVITY	A business activity is used to represent the state of the business process of one of the partners. For instance the requester is either in the state of sending the request, in the state of waiting for the response, or in the state of receiving.
BUSINESS COLLABORATION	An activity conducted between two or more parties for the purpose of achieving a specified outcome.
BUSINESS DOCUMENT	The set of information components that are interchanged as part of a business activity.
BUSINESS PARTNER	An entity that engages in business transactions with another business partner(s).
BUSINESS PROCESS	The means by which one or more activities are accomplished in operating business practices.
BUSINESS PROCESS SPECIFICATION SCHEMA	Defines the necessary set of elements to specify run-time aspects and configuration parameters to drive the partners' systems used in the collaboration. The goal of the BP Specification Schema is to provide the bridge between the eBusiness process modeling and specification of eBusiness software components.
BUSINESS TRANSACTION	A business transaction is a logical unit of business conducted by two or more parties that generates a computable success or failure state. The community, the partners, and the process, are all in a definable, and self-reliant state prior to the business transaction, and in a new definable, and self-reliant state after the business transaction. In other words if you are still 'waiting' for your business partner's response or reaction, the business transaction has not completed.
CLIENT	Software that initiates a connection with a *Server*.
COLLABORATION	Two or more parties working together under a defined set of rules.

COLLABORATION PROTOCOL	The protocol that defines for a Collaborative Process: 1. The sequence, dependencies and semantics of the Documents that are exchanged between Parties in order to carry out that Collaborative Process, and 2. The Messaging Capabilities used when sending documents between those Parties. Note that a Collaborative Process can have more than one Collaboration Protocol by which it can be implemented.
COLLABORATION PROTOCOL AGREEMENT (CPA)	Information agreed between two (or more) Parties that identifies or describes the specific Collaboration Protocol that they have agreed to use. A CPA indicates what the involved Parties "will" do when carrying out a Collaborative Process. A CPA is representable by a Document.
COLLABORATION PROTOCOL PROFILE (CPP)	Information about a Party that can be used to describe one or more Collaborative Processes and associated Collaborative Protocols that the Party supports. A CPP indicates what a Party "can" do in order to carry out a Collaborative Process. A CPP is representable by a Document. While logically, a CPP is a single document, in practice, the CPP might be a set of linked documents that express various aspects of the capabilities. A CPP is not an agreement. It represents the capabilities of a Party.
COLLABORATIVE PROCESS	A shared process by which two Parties work together in order to carry out a process. The Collaborative Process can be defined by an ebXML Collaboration Model.
CONFORMANCE	Fulfillment of a product, process or service of all requirements specified; adherence of an implementation to the requirements of one or more specific standards or technical specifications.
DIGITAL SIGNATURE	A digital code that can be attached to an electronically transmitted message that uniquely identifies the sender
DOCUMENT	A Document is any data that can be represented in a digital form.
DOCUMENT EXCHANGE	An exchange of documents between two parties.
ENCRYPTION	Cryptographic transformation of data (called "plaintext") into a form (called "ciphertext") that conceals the data's original meaning to prevent it from being known or used. If the transformation is reversible, the corresponding reversal process is called "decryption", which is a transformation that restores encrypted data to its original state.
EXTENSIBLE MARKUP LANGUAGE	XML is designed to enable the exchange of information (data) between different applications and data sources on the World Wide Web and has been standardized by the W3C.
IMPLEMENTATION	An implementation is the realization of a specification. It can be a software product, system or program.
MESSAGE	The movement of a document from one party to another.

MESSAGE HEADER	A specification of the structure and composition of the information necessary for an ebXML Messaging Service to successfully generate or process an ebXML compliant message.
MESSAGING CAPABILITIES	The set of capabilities that support exchange of Documents between Parties. Examples are the communication protocol and its parameters, security definitions, and general properties of sending and receiving messages.
MESSAGING SERVICE	A framework that enables interoperable, secure and reliable exchange of Messages between Trading Partners.
PACKAGE	A general-purpose mechanism for organizing elements into groups. Packages can be nested within other packages.
PARTY	A Party is an entity such as a company, department, organization or individual that can generate, send, receive or relay Documents.
PARTY DISCOVERY PROCESS	A Collaborative Process by which one Party can discover CPP information about other Parties.
PAYLOAD	A section of data/information that is not part of the ebXML wrapping.
PAYLOAD CONTAINER	A container used to envelope the real payload of an ebXML message. If a payload is present, the payload container consists of a MIME header portion (the ebXML Payload Envelope) and a content portion (the payload itself).
PAYLOAD ENVELOPE	The specific MIME headers that are associated with a MIME part.
RECEIVER	Recipient of a *Message*.
REGISTRY	A mechanism whereby relevant repository items and metadata about them can be registered such that a pointer to their location, and all their metadata, can be retrieved as a result of a query.
REQUESTER	Initiator of a *Business Transaction*.
RESPONDER	A counterpart to the initiator in a *Business Transaction*.
ROLE	The named specific behavior of an entity participating in a particular context. A role could be static (e.g., an association end) or dynamic (e.g., a collaboration role).
SECURITY POLICY	A set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources.
SENDER	Originator of a *Message*.
SERVER	Software that accepts a connection initiated by a *Client*.

UNIQUE IDENTIFIER	The abstract concept of utilizing a standard mechanism and process for assigning a sequence of alphanumeric codes to ebXML Registry items, including: Core Components, Aggregate Information Entities, and Business Processes.
UNIVERSALLY UNIQUE IDENTIFIER (UUID)	An identifier that is unique across both space and time, with respect to the space of all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.

7368
7369

ICS 35.040

Price based on 154 pages