



Creating A Single Global Electronic Market



1

2

3 **Collaboration-Protocol Profile and Agreement**

4 **Specification**

5 **Version 1.9**

6

7

8 **OASIS ebXML Collaboration Protocol Profile**

9 **and Agreement Technical Committee**

10

April 19, 2002

13 **1 Status of this Document**

14 This document specifies an ebXML SPECIFICATION for the eBusiness community.

15 Distribution of this document is unlimited.

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

21 ***This version:***

23 [http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-1\\_9.pdf](http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-1_9.pdf)

25 ***Errata for this version:***

27 [http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2\\_0-Errata.shtml](http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0-Errata.shtml)

29 ***Previous version:***

31 <http://www.ebxml.org/specs/ebCCP.pdf>

**33    2 Technical Committee Members**

34	Selem Aissi	Intel
35	Arvola Chan	TIBCO
36	James Bryce Clark	Individual Member
37	David Fischer	Drummond Group
38	Tony Fletcher	Individual Member
39	Brian Hayes	Commerce One
40	Neelakantan Kartha	Sterling Commerce
41	Kevin Liu	SAP
42	Pallavi Malu	Intel
43	Dale Moberg	Cyclone Commerce
44	Himagiri Mukkamala	Sybase
45	Peter Ogden	Cyclone Commerce
46	Marty Sachs	IBM
47	Yukinori Saito	Individual Member
48	David Smiley	Mercator
49	Tony Weida	Individual Member
50	Pete Wenzel	SeeBeyond
51	Jean Zheng	Vitria
52		

### 53   **3 ebXML Participants**

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

56

57   David Burdett, CommerceOne

58   Tim Chiou, United World Chinese Commercial Bank

59   Chris Ferris, Sun

60   Scott Hinkelmann, IBM

61   Maryann Hondo, IBM

62   Sam Hunting, ECOM XML

63   John Ibbotson, IBM

64   Kenji Itoh, JASTPRO

65   Ravi Kacker, eXcelon Corp.

66   Thomas Limanek, iPlanet

67   Daniel Ling, VCHEQ

68   Henry Lowe, OMG

69   Dale Moberg, Cyclone Commerce

70   Duane Nickull, XML Global Technologies

71   Stefano Pogliani, Sun

72   Rebecca Reed, Mercator

73   Karsten Riemer, Sun

74   Marty Sachs, IBM

75   Yukinori Saito, ECOM

76   Tony Weida, Edifecs

77

**78 4 Table of Contents**

79	1	Status of this Document.....	1
80	2	Technical Committee Members.....	2
81	3	ebXML Participants.....	3
82	4	Table of Contents.....	4
83	5	Introduction .....	8
84	5.1	Summary of Contents of Document .....	8
85	5.2	Document Conventions .....	8
86	5.3	Versioning of the Specification and Schema .....	9
87	5.4	Definitions .....	10
88	5.5	Audience.....	10
89	5.6	Assumptions .....	10
90	5.7	Related Documents.....	10
91	6	Design Objectives.....	11
92	7	System Overview.....	12
93	7.1	What This Specification Does .....	12
94	7.2	Forming a CPA from Two CPPs .....	14
95	7.3	Forming a CPA from a CPA Template .....	16
96	7.4	How the CPA Works .....	16
97	7.5	Where the CPA May Be Implemented .....	17
98	7.6	Definition and Scope .....	18
99	8	CPP Definition.....	19
100	8.1	Globally-Unique Identifier of CPP Instance Document .....	19
101	8.2	CPP Structure .....	20
102	8.3	CollaborationProtocolProfile element .....	20
103	8.4	PartyInfo Element.....	21
104	8.4.1	PartyId element.....	23
105	8.4.2	PartyRef element .....	24
106	8.4.2.1	xlink:type attribute.....	24
107	8.4.2.2	xlink:href attribute .....	24
108	8.4.2.3	type attribute .....	24
109	8.4.2.4	schemaLocation attribute .....	24
110	8.4.3	CollaborationRole element .....	25
111	8.4.4	ProcessSpecification element .....	27
112	8.4.4.1	name attribute .....	28
113	8.4.4.2	version attribute .....	28
114	8.4.4.3	xlink:type attribute .....	29
115	8.4.4.4	xlink:href attribute .....	29
116	8.4.4.5	uuid attribute .....	29
117	8.4.4.6	ds:Reference element .....	29
118	8.4.5	Role element .....	31
119	8.4.5.1	name attribute .....	31
120	8.4.5.2	xlink:type attribute .....	31
121	8.4.5.3	xlink:href attribute .....	31
122	8.4.6	ApplicationCertificateRef element .....	32
123	8.4.6.1	certId attribute .....	32
124	8.4.7	ApplicationSecurityDetailsRef element .....	32
125	8.4.7.1	SecurityId attribute .....	32
126	8.4.8	ServiceBinding element .....	32
127	8.4.9	Service element .....	33
128	8.4.9.1	type attribute .....	33
129	8.4.10	CanSend element .....	33
130	8.4.11	CanReceive element .....	34

131	8.4.12 ThisPartyActionBinding element.....	34
132	8.4.12.1 action attribute .....	35
133	8.4.12.2 packageId attribute.....	35
134	8.4.12.3 xlink:href attribute .....	35
135	8.4.12.4 xlink:type attribute .....	35
136	8.4.13 BusinessTransactionCharacteristics element.....	36
137	8.4.13.1 isNonRepudiationRequired attribute.....	36
138	8.4.13.2 isNonRepudiationReceiptRequired attribute .....	37
139	8.4.13.3 isSecureTransportRequired attribute.....	37
140	8.4.13.4 isConfidential attribute .....	37
141	8.4.13.5 isAuthenticated attribute .....	37
142	8.4.13.6 isAuthorizationRequired attribute.....	38
143	8.4.13.7 isTamperProof attribute .....	38
144	8.4.13.8 isIntelligibleCheckRequired attribute .....	38
145	8.4.13.9 timeToAcknowledgeReceipt attribute .....	38
146	8.4.13.10 timeToAcknowledgeAcceptance attribute .....	38
147	8.4.13.11 timeToPerform attribute .....	38
148	8.4.13.12 retryCount attribute.....	39
149	8.4.14 ChannelId element .....	39
150	8.4.15 ActionContext element .....	39
151	8.4.15.1 binaryCollaboration attribute .....	40
152	8.4.15.2 businessTransactionActivity attribute .....	40
153	8.4.15.3 requestOrResponseAction attribute .....	40
154	8.4.16 CollaborationActivity element.....	40
155	8.4.16.1 name attribute .....	41
156	8.4.17 Certificate element.....	41
157	8.4.17.1 certId attribute.....	41
158	8.4.17.2 ds:KeyInfo element.....	41
159	8.4.18 SecurityDetails element .....	41
160	8.4.18.1 securityId attribute .....	42
161	8.4.19 TrustAnchors element.....	42
162	8.4.20 SecurityPolicy element .....	43
163	8.4.21 DeliveryChannel element .....	43
164	8.4.21.1 channelId attribute .....	44
165	8.4.21.2 transportId attribute.....	44
166	8.4.21.3 docExchangeId attribute .....	44
167	8.4.22 MessagingCharacteristics element.....	44
168	8.4.22.1 syncReplyMode attribute .....	45
169	8.4.22.2 ackRequested attribute .....	46
170	8.4.22.3 ackSignatureRequested attribute .....	47
171	8.4.22.4 duplicateElimination attribute .....	47
172	8.4.22.5 actor attribute .....	48
173	8.4.23 Transport element .....	48
174	8.4.23.1 transportId attribute.....	49
175	8.4.24 TransportSender element .....	49
176	8.4.25 TransportProtocol element.....	49
177	8.4.26 AccessAuthentication element.....	50
178	8.4.27 TransportClientSecurity element .....	50
179	8.4.28 TransportSecurityProtocol element .....	50
180	8.4.29 ClientCertificateRef element .....	51
181	8.4.30 ServerSecurityDetailsRef element .....	51
182	8.4.31 Encryption Algorithm .....	51
183	8.4.32 TransportReceiver element .....	52
184	8.4.33 Endpoint element .....	52
185	8.4.33.1 uri attribute.....	52
186	8.4.33.2 type attribute .....	52

187	8.4.34 TransportServerSecurity element.....	53
188	8.4.35 ServerCertificateRef element.....	53
189	8.4.36 ClientSecurityDetailsRef element.....	53
190	8.4.37 Transport protocols.....	53
191	8.4.37.1 HTTP .....	53
192	8.4.37.2 SMTP .....	54
193	8.4.37.3 FTP .....	55
194	8.4.38 DocExchange Element.....	56
195	8.4.38.1 docExchangeId attribute .....	57
196	8.4.39 ebXMLSenderBinding element.....	57
197	8.4.39.1 version attribute .....	57
198	8.4.40 ReliableMessaging element.....	57
199	8.4.40.1 Retries and RetryInterval elements.....	58
200	8.4.40.2 MessageOrderSemantics element .....	58
201	8.4.41 PersistDuration element.....	58
202	8.4.42 SenderNonRepudiation element .....	58
203	8.4.43 NonRepudiationProtocol element.....	59
204	8.4.44 HashFunction element .....	59
205	8.4.45 SignatureAlgorithm element.....	59
206	8.4.45.1 oid attribute.....	60
207	8.4.45.2 w3c attribute .....	60
208	8.4.45.3 enumeratedType attribute .....	60
209	8.4.46 SigningCertificateRef element.....	60
210	8.4.47 SenderDigitalEnvelope element.....	60
211	8.4.48 DigitalEnvelopeProtocol element.....	61
212	8.4.49 EncryptionAlgorithm element .....	61
213	8.4.49.1 minimumStrength attribute .....	61
214	8.4.49.2 oid attribute .....	61
215	8.4.49.3 w3c attribute .....	61
216	8.4.49.4 enumeratedTypeAttribute .....	61
217	8.4.50 EncryptionSecurityDetailsRef element.....	62
218	8.4.51 NamespaceSupported element.....	62
219	8.4.52 ebXMLReceiverBinding element.....	62
220	8.4.53 ReceiverNonRepudiation element .....	63
221	8.4.54 SigningSecurityDetailsRef element.....	63
222	8.4.55 ReceiverDigitalEnvelope element .....	63
223	8.4.56 EncryptionCertificateRef element .....	64
224	8.4.57 OverrideMshActionBinding element.....	64
225	8.5 SimplePart element.....	64
226	8.6 Packaging element .....	65
227	8.6.1 ProcessingCapabilities element .....	66
228	8.6.2 CompositeList element .....	66
229	8.7 Signature element .....	68
230	8.8 Comment element.....	68
231	9 CPA Definition .....	70
232	9.1 CPA Structure.....	70
233	9.2 CollaborationProtocolAgreement element.....	71
234	9.3 Status Element .....	71
235	9.4 CPA Lifetime.....	72
236	9.4.1 Start element .....	72
237	9.4.2 End element .....	72
238	9.5 ConversationConstraints Element.....	73
239	9.5.1 invocationLimit attribute .....	73
240	9.5.2 concurrentConversations attribute .....	74
241	9.6 PartyInfo Element.....	74
242	9.6.1 ProcessSpecification element .....	74

243	9.7 SimplePart element.....	74
244	9.8 Packaging element.....	74
245	9.9 Signature element .....	75
246	9.9.1 Persistent Digital Signature .....	75
247	9.9.1.1 Signature Generation .....	75
248	9.9.1.2 ds:SignedInfo element .....	76
249	9.9.1.3 ds:CanonicalizationMethod element.....	76
250	9.9.1.4 ds:SignatureMethod element .....	76
251	9.9.1.5 ds:Reference element.....	76
252	9.9.1.6 ds:Transform element .....	76
253	9.9.1.7 ds:Algorithm attribute.....	77
254	9.10 Comment element.....	77
255	9.11 Composing a CPA from Two CPPs.....	77
256	9.11.1 ID Attribute Duplication.....	77
257	9.12 Modifying Parameters of the Process-Specification Document Based on Information in the CPA .....	78
258	10 References .....	79
259	11 Conformance .....	82
260	12 Disclaimer.....	83
261	13 Contact Information.....	84
262	Notices.....	86
263	Appendix A Example of CPP Document (Non-Normative).....	87
264	Appendix B Example of CPA Document (Non-Normative) .....	102
265	Appendix C Business Process Specification Corresponding to Complete CPP and CPA Definition (Non-Normative) .....	117
266	Appendix D W3C XML Schema Document Corresponding to Complete CPP and CPA Definition (Normative)...119	
267	Appendix E CPA Composition (Non-Normative).....	129
268	E.1 Suggestions for Design of Computational Procedures .....	129
269	E.2 CPA Formation Component Tasks.....	131
270	E.3 CPA Formation from <i>CPPs</i> : Context of Tasks .....	131
271	E.4 Business Collaboration Process Matching Tasks .....	132
272	E.5 Implementation Matching Tasks .....	133
273	E.6 CPA Formation: Technical Details .....	149
274	Appendix F Correspondence Between CPA and ebXML Messaging Parameters (Normative) .....	151
275	Appendix G Glossary of Terms .....	154
276		
277		

## 278    5 Introduction

### 280    5.1 Summary of Contents of Document

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

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

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

293    The rest of this specification is organized as follows:

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

### 313    5.2 Document Conventions

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

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

320 References to external documents are represented with BLOCK text enclosed in brackets, e.g.  
321 [RFC2396]. The references are listed in Section 10, "References".  
322

323 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD  
324 NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be  
325 interpreted as described in [RFC 2119].  
326

327 NOTE: Vendors SHOULD carefully consider support of elements with cardinalities (0 or  
328 1) or (0 or more). Support of such an element means that the element is processed  
329 appropriately for its defined function and not just recognized and ignored. A given *Party*  
330 might use these elements in some *CPPs* or *CPAs* and not in others. Some of these elements  
331 define parameters or operating modes and SHOULD be implemented by all vendors. It  
332 might be appropriate to implement elective elements that represent major run-time  
333 functions, such as various alternative communication protocols or security functions, by  
334 means of plug-ins so that a given *Party* MAY acquire only the needed functions rather than  
335 having to install all of them.  
336

337 By convention, values of [XML] attributes are generally enclosed in quotation marks, however  
338 those quotation marks are not part of the values themselves.  
339

### 340 **5.3 Versioning of the Specification and Schema**

341 Whenever this specification is modified, it SHALL be given a new version number.  
342

343 It is anticipated that during the review period, errors and inconsistencies in the specification and  
344 in the schema may be detected and have to be corrected. All known errors in the specification as  
345 well as necessary changes to the schema will be summarized in an errata page found at  
346

347 [http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2\\_0-Errata.shtml](http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0-Errata.shtml)  
348

349 The specification, when initially approved by the OASIS ebXML Collaboration Protocol Profile  
350 and Agreement Technical Committee for public review, SHALL carry a version number of  
351 "2\_0". At that time, the schema SHALL have a version number of "2\_0a" and the suffix letter  
352 after "2\_0" will be advanced as necessary when bug fixes to the schema have to be introduced.  
353 Such versions of the schema SHALL be found under the directory  
354

355 <http://www.oasis-open.org/committees/ebxml-cppa/schema/>  
356

357 In addition, the latest version of the schema SHALL always be found at  
358

359 [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)  
360

361 since the latter is the namespace URI used for this specification and the corresponding schema is  
362 supposed to be directly resolvable from the namespace URI.  
363

364 The value of the version attribute of the Schema element in a given version of the schema  
365 SHALL be equal to the version of the schema.  
366

367   **5.4 Definitions**

368   Technical terms in this specification are defined in Appendix G.

369

370   **5.5 Audience**

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

375

376   **5.6 Assumptions**

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

379

380   **5.7 Related Documents**

381   Related documents include ebXML Specifications on the following topics:

382

- 383       • ebXML Message Service Specification[ebMS]
- 384       • ebXML Business Process Specification Schema[ebBPSS]
- 385       • ebXML Core Component Overview[ccOVER]
- 386       • ebXML Registry Services Specification[ebRS]

387

388   See Section 10 for the complete list of references.

389

## 390 **6 Design Objectives**

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

400  
401 In addition to supporting direct interaction between two *Parties*, this specification MAY also be  
402 used to support interaction between two *Parties* through an intermediary such as a portal or  
403 broker.

404  
405 It is an objective of this specification that a *CPA* SHALL be capable of being composed by  
406 intersecting the respective *CPPs* of the *Parties* involved. The resulting *CPA* SHALL contain  
407 only those elements that are in common, or compatible, between the two *Parties*. Variable  
408 quantities, such as number of retries of errors, are then negotiated between the two *Parties*. The  
409 design of the *CPP* and *CPA* schemata facilitates this composition/negotiation process. However,  
410 the composition and negotiation processes themselves are outside the scope of this specification.  
411 Appendix E contains a non-normative discussion of this subject.

412  
413 It is a further objective of this specification to facilitate migration of both traditional EDI-based  
414 applications and other legacy applications to platforms based on the ebXML specifications. In  
415 particular, the *CPP* and *CPA* are components of the migration of applications based on the X12  
416 838 Trading-Partner Profile[X12] to more automated means of setting up *Business* relationships  
417 and doing *Business* under them.

## 418 7 System Overview

### 419 7.1 What This Specification Does

420 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.

424  
425 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 426 be expressed as a *Collaboration-Protocol Agreement (CPA)*.

428  
429 A *Party* MAY describe itself in a single *CPP*. A *Party* MAY create multiple *CPPs* that describe, 430 for example, different *Business Collaborations* that it supports, its operations in different regions 431 of the world, or different parts of its organization.

432  
433 To enable *Parties* wishing to do *Business* to find other *Parties* that are suitable *Business* 434 *Partners*, *CPPs* MAY be stored in a repository such as is provided by the ebXML 435 Registry[ebRS]. Using a discovery process provided as part of the specifications of a repository, 436 a *Party* MAY then use the facilities of the repository to find *Business Partners*.

437  
438 The document that defines the interactions between two *Parties* is a *Process-Specification* 439 document that MAY conform to the ebXML Business Process Specification Schema[ebBPSS]. 440 The *CPP* and *CPA* include references to this *Process-Specification* document. The *Process-* 441 *Specification* document MAY be stored in a repository such as the ebXML Registry. See NOTE 442 about alternative *Business-Collaboration* descriptions in Section 8.4.4.

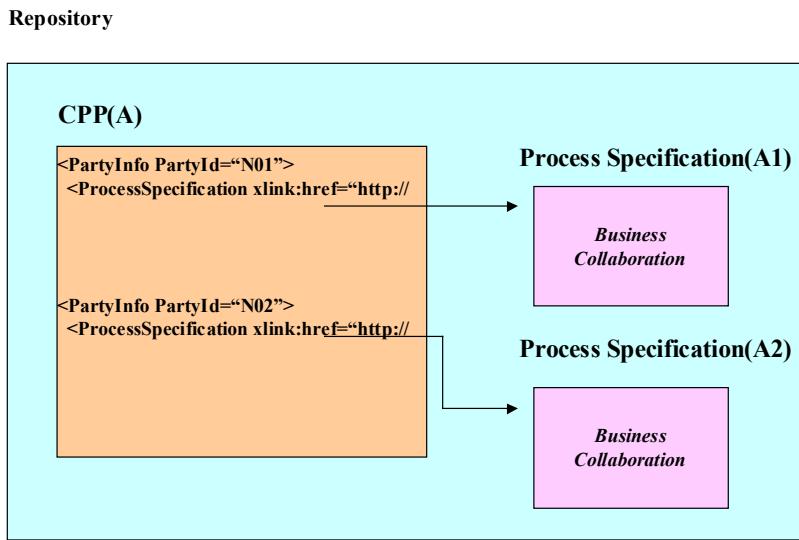
443  
444 Figure 1 illustrates the relationships between a *CPP* and two *Process-Specification* documents, 445 A1 and A2, in an ebXML Registry. On the left is a *CPP*, A, which includes information about 446 two parts of an enterprise that are represented as different *Parties*. On the right are shown two 447 *Process-Specification* documents. Each of the *PartyInfo* elements in the *CPP* contains a 448 reference to one of the *Process-Specification* documents. This identifies the *Business* 449 *Collaboration* that the *Party* can perform.

450  
451 This specification defines the markup language vocabulary for creating electronic *CPPs* and 452 *CPAs*. *CPPs* and *CPAs* are [XML] documents. In the appendices of this specification are two 453 sample *CPPs*, a sample *CPA* formed from the *CPPs*, a sample *Process-Specification* referenced 454 by the *CPPs* and the *CPA*, and the XML Schema governing the structures of *CPPs* and *CPAs*.

455  
456 The *CPP* describes the capabilities of an individual *Party*. A *CPA* describes the capabilities that 457 two *Parties* have agreed to use to perform a particular *Business Collaboration*. These *CPAs* 458 define the "information technology terms and conditions" that enable *Business* documents to be 459 electronically interchanged between *Parties*. The information content of a *CPA* is similar to the 460 information-technology specifications sometimes included in Electronic Data Interchange (EDI) 461 *Trading Partner Agreements (TPAs)*. However, these *CPAs* are not paper documents. Rather,

462 they are electronic documents that can be processed by computers at the *Parties'* sites in order to  
 463 set up and then execute the desired *Business* information exchanges. The "legal" terms and  
 464 conditions of a *Business* agreement are outside the scope of this specification and therefore are  
 465 not included in the *CPP* and *CPA*.

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



466  
 467 An enterprise MAY choose to represent itself as multiple *Parties*. For example, it might  
 468 represent a central office supply procurement organization and a manufacturing supplies  
 469 procurement organization as separate *Parties*. The enterprise MAY then construct a *CPP* that  
 470 includes all of its units that are represented as separate *Parties*. In the *CPP*, each of those units  
 471 would be represented by a separate *PartyInfo* element.  
 472

473 The CPPA specification is concerned with software that conducts business on behalf of *Parties*  
 474 by exchanging *Messages*[ebMS]. In particular, it is concerned with *Client* and *Server* software  
 475 programs that engage in *Business Transactions*[ebBPSS] by sending and receiving *Messages*.  
 476 Those *Messages* convey *Business Documents* and/or business signals[ebBPSS] in their payload.  
 477 Under the terms of a *CPA*:

- 478
- 479     • A *Client* initiates a connection with a *Server*.
  - 480     • A *Requester* initiates a *Business Transaction* with a *Responder*.
  - 481     • A *Sender* sends a *Message* to a *Receiver*.

482  
 483 Thus, *Client* and *Server* are software counterparts, *Requester* and *Responder* are business  
 484 counterparts, and *Sender* and *Receiver* are messaging counterparts. There is no fixed  
 485 relationship between counterparts of different types. For example, consider a purchasing  
 486 collaboration. *Client* software representing the buying party might connect to *Server* software

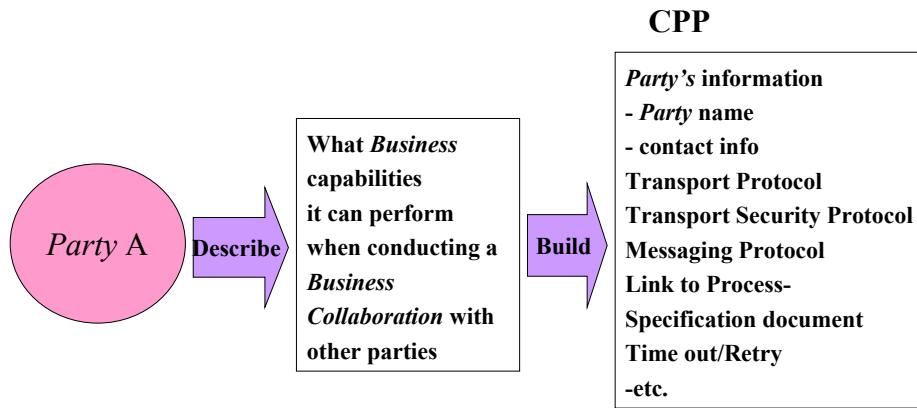
487 representing the selling party, and then make a purchase request by sending a *Message*  
488 containing a purchase order over that connection. If the CPA specifies a synchronous business  
489 response, the *Server* might then respond by sending a *Message* containing an acceptance notice  
490 back to the *Client* over the same connection. Alternatively, if the CPA specifies an  
491 asynchronous business response, *Client* software representing the selling party might later  
492 respond by connecting to *Server* software representing the buying party and then sending a  
493 *Message* containing an acceptance notice.

494  
495 In general, the *Parties* to a *CPA* can have both client and server characteristics. A client requests  
496 services and a server provides services to the *Party* requesting services. In some applications,  
497 one *Party* only requests services and one *Party* only provides services. These applications have  
498 some resemblance to traditional client-server applications. In other applications, each *Party*  
499 MAY request services of the other. In that case, the relationship between the two *Parties* can be  
500 described as a peer-peer relationship rather than a client-server relationship.  
501

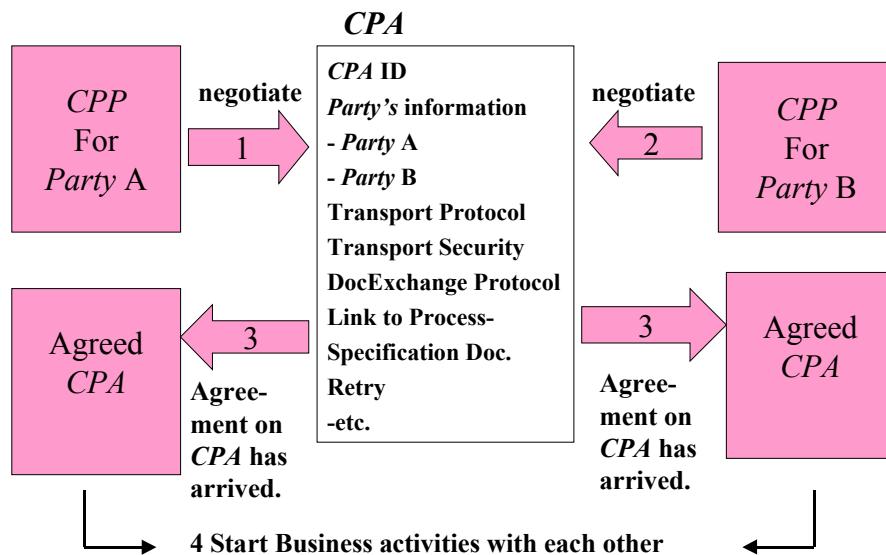
## 502 **7.2 Forming a CPA from Two CPPs**

503 This section summarizes the process of discovering a *Party* to do *Business* with and forming a  
504 *CPA* from the two *Parties' CPPs*. In general, this section is an overview of a possible procedure  
505 and is not to be considered a normative specification. See Appendix E "CPA Composition (Non-  
506 Normative)" for more information.  
507

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

**Figure 2: Overview of Collaboration-Protocol Profiles (CPP)**514  
515  
516  
517

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

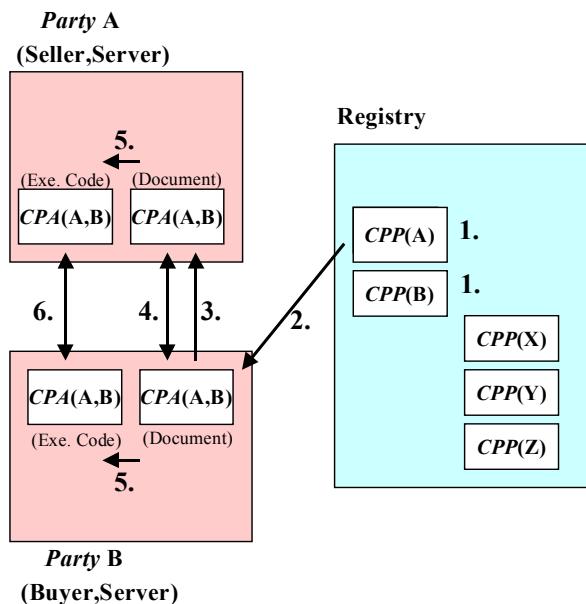
**Figure 3: Overview of Collaboration-Protocol Agreements (CPA)**

518

519 Figure 4 illustrates the entire process. The steps are listed at the left. The end of the process is  
 520 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**

1. Any *Party* may register its CPPs to an ebXML Registry.
2. *Party B* discovers trading partner A (Seller) by searching in the Registry and downloads CPP(A) to *Party B*'s server.
3. *Party B* creates CPA(A,B) and sends CPA(A,B) to *Party A*.
4. *Parties A and B* negotiate and store identical copies of the completed *CPA* as a document in both servers. This process is done manually or automatically.
5. *Parties A and B* configure their run-time systems with the information in the *CPA*.
6. *Parties A and B* do business under the new *CPA*.



521 then ready to do *Business*.

522

### 523 7.3 Forming a CPA from a CPA Template

524 Alternatively, a *CPA* template might be used to create a *CPA*. A *CPA* template represents one  
 525 party's "fill in the blanks" proposal to a prospective trading partner for implementing one or  
 526 more business processes. For example, such a template might contain placeholder values for  
 527 identifying aspects of the other party. To form a *CPA* from a *CPA* template, the placeholder  
 528 values would be replaced by the actual values for the other trading partner. Actual values might  
 529 be obtained from the other party's CPP, if available, or by data entry in an HTML form, among  
 530 other possibilities. The current version of this specification does not address how placeholder  
 531 values might be represented in a *CPA*. However, the process of filling out a *CPA* template  
 532 MUST result in a valid *CPA*. Further discussion of *CPA* templates is provided in Appendix E.  
 533

### 534 7.4 How the CPA Works

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

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

542  
543 The information in the *CPA* is used to configure the *Parties'* systems to enable exchange of  
544 *Messages* in the course of performing the selected *Business Collaboration*. Typically, the  
545 software that performs the *Message* exchanges and otherwise supports the interactions between  
546 the *Parties* is middleware that can support any selected *Business Collaboration*. One component  
547 of this middleware MAY be the ebXML Message Service Handler[ebMS]. In this specification,  
548 the term "run-time system" or "run-time software" is used to denote such middleware.

549  
550 The *CPA* and the *Process-Specification* document that it references define a conversation  
551 between the two *Parties*. The conversation represents a single unit of *Business* as defined by the  
552 *Binary-Collaboration* component of the *Process-Specification* document. The conversation  
553 consists of one or more *Business Transactions*, each of which is a request *Message* from one  
554 *Party* and zero or one response *Message* from the other *Party*. The *Process-Specification*  
555 document defines, among other things, the request and response *Messages* for each *Business*  
556 *Transaction* and the order in which the *Business Transactions* are REQUIRED to occur. See  
557 [ebBPSS] for a detailed explanation.

558  
559 The *CPA* MAY actually reference more than one *Process-Specification* document. When a *CPA*  
560 references more than one *Process-Specification* document, each *Process-Specification* document  
561 defines a distinct type of conversation. Any one conversation involves only a single *Process-*  
562 *Specification* document.

563  
564 A new conversation is started each time a new unit of *Business* is started. The *Business*  
565 *Collaboration* also determines when the conversation ends. From the viewpoint of a *CPA*  
566 between *Party A* and *Party B*, the conversation starts at *Party A* when *Party A* sends the first  
567 request *Message* to *Party B*. At *Party B*, the conversation starts when it receives the first request  
568 of the unit of *Business* from *Party A*. A conversation ends when the *Parties* have completed the  
569 unit of *Business*.

570  
571 NOTE: The run-time system SHOULD provide an interface by which the *Business*  
572 application can request initiation and ending of conversations.  
573

## 574 **7.5 Where the CPA May Be Implemented**

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

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

586  
587     NOTE: It is possible to provide a graphical *CPP/CPA*-authoring tool that understands both  
588     the semantics of the *CPP/CPA* and the XML syntax. Equally important, the definitions in  
589     this specification make it feasible to automatically generate, at each *Party's* site, the code  
590     needed to execute the *CPA*, enforce its rules, and interface with the *Party's* back-end  
591     processes.  
592

## 593 **7.6 Definition and Scope**

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

600  
601     NOTE: This specification is limited to defining the contents of the *CPP* and *CPA*, and it is  
602     possible to be conformant with it merely by producing a *CPP* or *CPA* document that  
603     conforms to the XML Schema document defined herein. It is, however, important to  
604     understand that the value of this specification lies in its enabling a run-time system that  
605     supports electronic commerce between two *Parties* under the guidance of the information in  
606     the *CPA*.

## 607 8 CPP Definition

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

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

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

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

622 **Delivery Channels** - A delivery channel describes a *Party's* *Message*-receiving and *Message*-  
623 sending characteristics. It consists of one document-exchange definition and one transport  
624 definition. Several delivery channels MAY be defined in one *CPP*.

625 **Document-Exchange Layer** - The Document-exchange layer specifies processing of the  
626 business documents by the Message-exchange function. Properties specified include encryption,  
627 digital signature, and reliable-messaging characteristics. The options selected for the Document-  
628 exchange layer are complementary to those selected for the transport layer. For example, if  
629 Message security is desired and the selected transport protocol does not provide *Message*  
630 encryption, then *Message* encryption MUST be specified in the Document-exchange layer. The  
631 protocol for exchanging *Messages* between two *Parties* is defined by the ebXML Message  
632 Service specification[ebMS] or other similar messaging services.

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

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

### 640 8.1 Globally-Unique Identifier of CPP Instance Document

641 When a *CPP* is placed in an ebXML or other Registry, the Registry assigns it a globally unique  
642 identifier (GUID) that is part of its metadata. That GUID MAY be used to distinguish among  
643 *Collaboration-Protocol Profile and Agreement Specification*

651    *CPPs* belonging to the same *Party*.

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

## 657    8.2 CPP Structure

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

```
660      <tp:CollaborationProtocolProfile  
661          xmlns:tp="http://www.oasis-open.org/committees/ebxml-  
662          cппa/schema/cpp-cpa-2_0.xsd"  
663          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
664          xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-  
665          cппa/schema/cpp-cpa-2_0.xsd http://www.oasis-open.org/committees/ebxml-  
666          cппa/schema/cpp-cpa-2_0.xsd"  
667          xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
668          xmlns:xlink="http://www.w3.org/1999/xlink"  
669          tp:cpid="uri:companyA-cpp"  
670          tp:version="2_0a">  
671          <tp:PartyInfo> <!-- one or more -->  
672          ...  
673          </tp:PartyInfo>  
674          <tp:SimplePart id="..."><!-- one or more -->  
675          ...  
676          </tp:SimplePart>  
677          <tp:Packaging id="..."><!-- one or more -->  
678          ...  
679          </tp:Packaging>  
680          <tp:Signature> <!-- zero or one -->  
681          ...  
682          </tp:Signature>  
683          <tp:Comment>text</tp:Comment> <!-- zero or more -->  
684      </tp:CollaborationProtocolProfile>
```

## 687    8.3 CollaborationProtocolProfile element

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

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

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

696    In addition, the **CollaborationProtocolProfile** element contains a REQUIRED *cpid* attribute  
697    that supplies a unique identifier for the document, plus a REQUIRED *version* attribute that  
698    indicates the version of the schema. Its purpose is to identify the version of the schema that the  
699    *CPP* conforms to. The value of the *version* attribute SHOULD be a string such as "2\_0a",  
700    "2\_0b", etc.

702  
703       NOTE: The method of assigning unique cppid values is left to the implementation.  
704

705       The **CollaborationProtocolProfile** element SHALL consist of the following child elements:  
706

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

715  
716       A *CPP* document MAY be digitally signed so as to provide for a means of ensuring that the  
717       document has not been altered (integrity) and to provide for a means of authenticating the author  
718       of the document. A digitally signed *CPP* SHALL be signed using technology that conforms to  
719       the joint W3C/IETF XML Digital Signature specification[XMLDSIG].  
720

## 721       **8.4 PartyInfo Element**

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

727  
728       <tp:PartyInfo  
729            tp:partyName="..."  
730            tp:defaultMshChannelId="..."  
731            tp:defaultMshPackageId="...">  
732            <tp:PartyId tp:type="..."><!-- one or more -->  
733            ...  
734            </tp:PartyId>  
735            <tp:PartyRef xlink:href="..."/>  
736            <tp:CollaborationRole> <!-- one or more -->  
737            ...  
738            </tp:CollaborationRole>  
739            <tp:Certificate> <!-- one or more -->  
740            ...  
741            </tp:Certificate>  
742            <tp:SecurityDetails> <!-- one or more -->  
743            ...  
744            </tp:SecurityDetails>  
745            <tp:DeliveryChannel> <!-- one or more -->  
746            ...  
747            </tp:DeliveryChannel>  
748            <tp:Transport> <!-- one or more -->  
749            ...  
750            </tp:Transport>  
751            <tp:DocExchange> <!-- one or more -->  
752            ...  
753            </tp:DocExchange>  
754            </tp:OverrideMshActionBinding> <!-- zero or more -->

```
755 ...
756     </tp:OverrideMshActionBinding>
757   </tp:PartyInfo>
758
```

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

764 The following example illustrates two possible party names.

```
765
766     <tp:PartyInfo tp:partyName="Example, Inc." ...></tp:PartyInfo>
767
768     <tp:PartyInfo tp:partyName="Example, Inc. US Western Division">
769     ...
770   </tp:PartyInfo>
771
772
```

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

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

- 783 • One or more REQUIRED **PartyId** elements that provide logical identifiers for the  
784 organization.
- 785 • One or more REQUIRED **PartyRef** elements that provide pointers to more  
786 information about the *Party*.
- 787 • One or more REQUIRED **CollaborationRole** elements that identify the roles that this  
788 *Party* can play in the context of a *Process Specification*.
- 789 • One or more REQUIRED **Certificate** elements that identify the certificates used by  
790 this *Party* in security functions.
- 791 • One or more REQUIRED **SecurityDetails** elements that identify trust anchors and  
792 specify security policy used by this *Party* in security functions.
- 793 • One or more REQUIRED **DeliveryChannel** elements that define the characteristics  
794 that the *Party* can use to send and/or receive *Messages*. It includes both the transport  
795 protocol (e.g. HTTP) and the messaging protocol (e.g. ebXML *Message Service*).
- 796 • One or more REQUIRED **Transport** elements that define the characteristics of the  
797 transport protocol(s) that the *Party* can support to send and/or receive *Messages*.
- 798 • One or more REQUIRED **DocExchange** elements that define the *Message-exchange*  
799 characteristics, such as the signature and encryption protocols, that the *Party* can  
800 support.
- 801 • Zero or more **OverrideMshActionBinding** elements that specify the *DeliveryChannel*

803 to use for asynchronously delivered *Message Service Handler* level messages.  
804

805 **8.4.1 PartyId element**

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

811  
812 In a *CPP* that contains multiple **PartyInfo** elements, different **PartyInfo** elements MAY contain  
813 **PartyId** elements that define different logical identifiers. This permits a large organization, for  
814 example, to have different identifiers for different purposes.

815  
816 The value of the **PartyId** element is any string that provides a unique identifier. The identifier  
817 MAY be any identifier that is understood by both *Parties* to a *CPA*. Typically, the identifier  
818 would be listed in a well-known directory such as DUNS (Dun and Bradstreet) or in any naming  
819 system specified by [ISO6523].

820  
821 The **PartyId** element has a single IMPLIED attribute: **type** that has an anyURI [XMLSCHEMA-  
822 2] value.

823  
824 If the **type** attribute is present, then it provides a scope or namespace for the content of the  
825 **PartyId** element.

826  
827 If the **type** attribute is not present, the content of the **PartyId** element MUST be a URI that  
828 conforms to [RFC2396]. It is RECOMMENDED that the value of the **type** attribute be a URN  
829 that defines a namespace for the value of the **PartyId** element. Typically, the URN would be  
830 registered in a well-known directory of organization identifiers.

831  
832 The following example illustrates two URI references.

833  
834 <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-  
835 type:duns">123456789</tp:PartyId>  
836  
837 <tp:PartyId>urn:icann:example.com</tp:PartyId>

838  
839 The first example is the *Party's* DUNS number. The value is the DUNS number of the  
840 organization.

841  
842 The second example shows an arbitrary URN. This might be a URN that the *Party* has  
843 registered with IANA, the Internet Assigned Numbers Authority (<http://www.iana.org>) to  
844 identify itself directly.

845  
846 The following document discusses naming agencies and how they are identified via URI values  
847 of the **type** attribute:

848  
849 [http://www.oasis-open.org/committees/ebxml-cppa/documents/PartyID\\_Types.shtml](http://www.oasis-open.org/committees/ebxml-cppa/documents/PartyID_Types.shtml)

850

#### 851 8.4.2 PartyRef element

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

862

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

- 864 • a FIXED **xlink:type** attribute,
- 865 • a REQUIRED **xlink:href** attribute,
- 866 • an IMPLIED **type** attribute,
- 867 • an IMPLIED **schemaLocation** attribute.

868

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

872

##### 873 8.4.2.1 **xlink:type** attribute

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

876

##### 877 8.4.2.2 **xlink:href** attribute

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

880

##### 881 8.4.2.3 **type** attribute

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

886

##### 887 8.4.2.4 **schemaLocation** attribute

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

890

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

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

897

#### 898 8.4.3 CollaborationRole element

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

```
906
907     <tp:CollaborationRole tp:id="BuyerId">
908         <tp:ProcessSpecification
909             tp:version="2.0"
910             tp:name="PIP3A4RequestPurchaseOrder"
911             xlink:type="simple"
912             xlink:href="http://www.rosettanet.org/processes/3A4.xml"/>
913         <tp:Role
914             tp:name="Buyer"
915             xlink:type="simple"
916
917             xlink:href="http://www.rosettanet.org/processes/3A4.xml#BuyerId"/>
918             <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert" />
919             <tp:ServiceBinding>
920                 <tp:Service
921                     tp:type="anyURI">urn::icann:rosettanet.org:bpid:3A4$2.0</tp:Service>
922                     <tp:CanSend>
923                         <tp:ThisPartyActionBinding
924                             tp:id="companyA_ABID1"
925                             tp:action="Purchase Order Request Action"
926                             tp:packageId="CompanyA_RequestPackage">
927                             <tp:BusinessTransactionCharacteristics
928                                 tp:isNonRepudiationRequired="true"
929                                 tp:isNonRepudiationReceiptRequired="true"
930                                 tp:isSecureTransportRequired="true"
931                                 tp:isConfidential="transient"
932                                 tp:isAuthenticated="persistent"
933                                 tp:isTamperProof="persistent"
934                                 tp:isAuthorizationRequired="true"
935                                 tp:timeToAcknowledgeReceipt="PT2H"
936                                 tp:timeToPerform="P1D"/>
937                             <tp:ActionContext
938                                 tp:binaryCollaboration="Request Purchase Order"
939                                 tp:businessTransactionActivity="Request Purchase
940                                     Order"
941                                 tp:requestOrResponseAction="Purchase Order Request
942                                     Action"/>
943                                 <tp:ChannelId>asyncChannelA1</tp:ChannelId>
944                                 </tp:ThisPartyActionBinding>
945                         </tp:CanSend>
946                         <tp:CanSend>
947                             <tp:ThisPartyActionBinding
948                                 tp:id="companyA_ABID2"
949                                 tp:action="ReceiptAcknowledgment"
950                                 tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
951                                 <tp:BusinessTransactionCharacteristics
952                                     tp:isNonRepudiationRequired="true"
953                                     tp:isNonRepudiationReceiptRequired="true"
954                                     tp:isSecureTransportRequired="true"
955                                     tp:isConfidential="transient"
956                                     tp:isAuthenticated="persistent"
```

```
957                     tp:isTamperProof="persistent"
958                     tp:isAuthorizationRequired="true"
959                     tp:timeToAcknowledgeReceipt="PT2H"
960                     tp:timeToPerform="P1D"/>
961             <tp:ChannelId>asyncChannelA1</tp:ChannelId>
962         </tp:ThisPartyActionBinding>
963     </tp:CanSend>
964     <tp:CanReceive>
965         <tp:ThisPartyActionBinding
966             tp:id="companyA_ABID3"
967             tp:action="Purchase Order Confirmation Action"
968             tp:packageId="CompanyA_ResponsePackage">
969             <tp:BusinessTransactionCharacteristics
970                 tp:isNonRepudiationRequired="true"
971                 tp:isNonRepudiationReceiptRequired="true"
972                 tp:isSecureTransportRequired="true"
973                 tp:isConfidential="transient"
974                 tp:isAuthenticated="persistent"
975                 tp:isTamperProof="persistent"
976                 tp:isAuthorizationRequired="true"
977                 tp:timeToAcknowledgeReceipt="PT2H"
978                 tp:timeToPerform="P1D"/>
979             <tp:ActionContext
980                 tp:binaryCollaboration="Request Purchase Order"
981                 tp:businessTransactionActivity="Request Purchase
982 Order"
983                 tp:requestOrResponseAction="Purchase Order
984 Confirmation Action"/>
985             <tp:ChannelId>asyncChannelA1</tp:ChannelId>
986         </tp:ThisPartyActionBinding>
987     </tp:CanReceive>
988     <tp:CanReceive>
989         <tp:ThisPartyActionBinding
990             tp:id="companyA_ABID4"
991             tp:action="ReceiptAcknowledgment"
992             tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
993             <tp:BusinessTransactionCharacteristics
994                 tp:isNonRepudiationRequired="true"
995                 tp:isNonRepudiationReceiptRequired="true"
996                 tp:isSecureTransportRequired="true"
997                 tp:isConfidential="transient"
998                 tp:isAuthenticated="persistent"
999                 tp:isTamperProof="persistent"
1000                tp:isAuthorizationRequired="true"
1001                tp:timeToAcknowledgeReceipt="PT2H"
1002                tp:timeToPerform="P1D"/>
1003            <tp:ChannelId>asyncChannelA1</tp:ChannelId>
1004        </tp:ThisPartyActionBinding>
1005    </tp:CanReceive>
1006    <tp:CanReceive>
1007        <tp:ThisPartyActionBinding
1008            tp:id="companyA_ABID5"
1009            tp:action="Exception"
1010            tp:packageId="CompanyA_ExceptionPackage">
1011            <tp:BusinessTransactionCharacteristics
1012                tp:isNonRepudiationRequired="true"
1013                tp:isNonRepudiationReceiptRequired="true"
1014                tp:isSecureTransportRequired="true"
1015                tp:isConfidential="transient"
1016                tp:isAuthenticated="persistent"
1017                tp:isTamperProof="persistent"
1018                tp:isAuthorizationRequired="true"
1019                tp:timeToAcknowledgeReceipt="PT2H"
1020                tp:timeToPerform="P1D"/>
```

```
1021           <tp:ChannelId>asyncChannelA1</tp:ChannelId>
1022           </tp:ThisPartyActionBinding>
1023       </tp:CanReceive>
1024   </tp:ServiceBinding>
1025 </tp:CollaborationRole>
```

1027 To indicate that the *Party* can play roles in more than one *Business Collaboration* or more than  
1028 one role in a given *Business Collaboration*, the **PartyInfo** element SHALL contain more than  
1029 one **CollaborationRole** element. Each **CollaborationRole** element SHALL contain the  
1030 appropriate combination of **ProcessSpecification** element and **Role** element.

1031  
1032 The **CollaborationRole** element SHALL consist of the following child elements: a REQUIRED  
1033 **ProcessSpecification** element, a REQUIRED **Role** element, zero or more  
1034 **ApplicationCertificateRef** elements, zero or one **ApplicationSecurityDetailsRef** element, and  
1035 one **ServiceBinding** element. The **ProcessSpecification** element identifies the *Process-*  
1036 *Specification* document that defines such role. The **Role** element identifies which role the *Party*  
1037 is capable of supporting. The **ApplicationCertificateRef** element identifies the certificate to be  
1038 used for application level signature and encryption. The **ApplicationSecurityDetailsRef** element  
1039 identifies the trust anchors and security policy that will be applied to any application-level  
1040 certificate offered by the other *Party*. The **ServiceBinding** element SHALL consist of zero or  
1041 more **CanSend** elements and zero or more **CanReceive** elements. The **CanSend** and **CanReceive**  
1042 elements identify the **DeliveryChannel** elements that are to be used for sending and receiving  
1043 business action messages by the **Role** in question. They MAY also be used for specifying  
1044 **DeliveryChannels** for business signal messages.

1045  
1046 Each *Party* SHALL have a default delivery channel for the delivery of standalone *Message*  
1047 Service Handler level signals like (Reliable Messaging) Acknowledgments, Errors,  
1048 StatusRequest, StatusResponse, etc.

#### 1050 8.4.4 ProcessSpecification element

1051 The **ProcessSpecification** element provides the link to the *Process-Specification* document that  
1052 defines the interactions between the two *Parties*. It is RECOMMENDED that this *Business-*  
1053 *Collaboration* description be prepared in accordance with the ebXML Business Process  
1054 Specification Schema[ebBPSS]. The *Process-Specification* document MAY be kept in an  
1055 ebXML Registry.

1056  
1057 NOTE: A *Party* can describe the *Business Collaboration* using any desired alternative to  
1058 the ebXML Business Process Specification Schema. When an alternative *Business-*  
1059 *Collaboration* description is used, the *Parties* to a *CPA* MUST agree on how to interpret  
1060 the *Business-Collaboration* description and how to interpret the elements in the *CPA* that  
1061 reference information in the *Business-Collaboration* description. The affected elements  
1062 in the *CPA* are the **Role** element, the **CanSend** and **CanReceive** elements, the  
1063 **ActionContext** element, and some attributes of the **BusinessTransactionCharacteristics**  
1064 element.

1065  
1066 The syntax of the **ProcessSpecification** element is:

1067

```

1068      <tp:ProcessSpecification
1069          tp:version="2.0"
1070          tp:name="PIP3A4RequestPurchaseOrder"
1071          xlink:type="simple"
1072          xlink:href="http://www.rosettanet.org/processes/3A4.xml"
1073          uuid="urn:icann:rosettanet.org:bpid:3A4$2.0">
1074          <ds:Reference ds:URI="http://www.rosettanet.org/processes/3A4.xml">
1075              <ds:Transforms>
1076                  <ds:Transform
1077          ds:Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
1078      </ds:Transforms>
1079      <ds:DigestMethod
1080          ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1081          <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
1082      </ds:Reference>
1083  </tp:ProcessSpecification>
1084

```

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 **ProcessSpecifications**. Essentially, **ds:Reference** elements MUST be provided to correspond to the transitive closure of all **ProcessSpecifications** 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).

1118

**8.4.4.3 xlink:type attribute**

1119 The *xlink:type* attribute has a FIXED value of "simple". This identifies the element as being an  
1120 [XLINK] simple link.

1121

**8.4.4.4 xlink:href attribute**

1122 The REQUIRED *xlink:href* attribute SHALL have a value that identifies the *Process-*  
1123 *Specification* document and is a URI that conforms to [RFC2396].

1124

**8.4.4.5 uuid attribute**

1125 The IMPLIED *uuid* attribute uniquely identifies the *ProcessSpecification*. If the *Process-*  
1126 *Specification* document is defined by the ebXML Business Process specification [ebBPSS], then  
1127 this attribute MUST be set to the *uuid* for the corresponding *ProcessSpecification* element  
1128 within the business process specification instance.

1129

**8.4.4.6 ds:Reference element**

1130 The *ds:Reference* element identifies the same *Process-Specification* document as the enclosing  
1131 *ProcessSpecification* element's *xlink:href* attribute and additionally provides for verification that  
1132 the *Process-Specification* document has not changed since the *CPP* was created, through the use  
1133 of a digest method and digest value as described below.

1134

1135       NOTE: *Parties* MAY test the validity of the *CPP* or *CPA* at any time. The following  
1136       validity tests MAY be of particular interest:

1137

- 1138       • test of the validity of a *CPP* and the referenced *Process-Specification* documents at  
1139       the time composition of a *CPA* begins in case they have changed since they were  
1140       created,
- 1141       • test of the validity of a *CPA* and the referenced *Process-Specification* documents at  
1142       the time a *CPA* is installed into a *Party's* system,
- 1143       • test of the validity of a *CPA* at intervals after the *CPA* has been installed into a *Party's*  
1144       system. The *CPA* and the referenced *Process-Specification* documents MAY be  
1145       processed by an installation tool into a form suited to the particular middleware.  
1146       Therefore, alterations to the *CPA* and the referenced *Process-Specification* documents  
1147       do not necessarily affect ongoing run-time operations. Such alterations might not be  
1148       detected until it becomes necessary to reinstall the *CPA* and the referenced *Process-*  
1149       *Specification* documents.

1150

1151 The syntax and semantics of the *ds:Reference* element and its child elements are defined in the  
1152 XML Digital Signature specification[XMLDSIG]. In addition, to identify the *Process-*  
1153 *Specification* document, the first *ds:Reference* element MUST include a *ds:URI* attribute whose  
1154 value is identical to that of the *xlink:href* attribute in the enclosing *ProcessSpecification*  
1155 element.

1156

1157 According to [XMLDSIG], a *ds:Reference* element can have a *ds:Transforms* child element,  
1158 which in turn has an ordered list of one or more *ds:Transform* child elements to specify a  
1159 sequence of transforms. However, this specification currently REQUIRES the Canonical

1164 XML[XMLC14N] transform and forbids other transforms. Therefore, the following additional  
1165 requirements apply to a **ds:Reference** element within a **ProcessSpecification** element:

- 1166
- 1167 • The **ds:Reference** element MUST have a **ds:Transforms** child element.
  - 1168 • That **ds:Transforms** element MUST have exactly one **ds:Transform** child element.
  - 1169 • That **ds:Transform** element MUST specify the Canonical XML[XMLC14N]  
1170 transform via the following REQUIRED value for its REQUIRED **ds:Algorithm**  
1171 attribute: <http://www.w3.org/TR/2001/Rec-xml-c14n-20010315>.

1172

1173 Note that implementation of Canonical XML is REQUIRED by the XML Digital  
1174 Signature specification[XMLDSIG].

1175

1176 To enable verification that the identified and transformed *Process-Specification* document has  
1177 not changed, the **ds:DigestMethod** element specifies the digest algorithm applied to the *Process-*  
1178 *Specification* document, and the **ds:DigestValue** element specifies the expected value. The  
1179 *Process-Specification* document is presumed to be unchanged if and only if the result of applying  
1180 the digest algorithm to the *Process-Specification* document results in the expected value.

1181

1182 A **ds:Reference** element in a **ProcessSpecification** element has implications for *CPP* validity:

- 1183
- 1184 • A *CPP* MUST be considered invalid if any **ds:Reference** element within a  
1185 **ProcessSpecification** element fails reference validation as defined by the XML Digital  
1186 Signature specification[XMLDSIG].
  - 1187
  - 1188 • A *CPP* MUST be considered invalid if any **ds:Reference** element within it cannot be  
1189 dereferenced.

1190

1191 Other validity implications of such **ds:Reference** elements are specified in the description of the  
1192 **Signature** element in Section 9.9.

1193

1194 NOTE: The XML Digital Signature specification[XMLDSIG] states "The signature  
1195 application MAY rely upon the identification (URI) and Transforms provided by the  
1196 signer in the Reference element, or it MAY obtain the content through other means such  
1197 as a local cache" (emphasis on MAY added). However, it is RECOMMENDED that  
1198 ebXML *CPP/CPA* implementations not make use of such cached results when signing or  
1199 validating.

1200

1201 NOTE: It is recognized that the XML Digital Signature specification[XMLDSIG]  
1202 provides for signing an XML document together with externally referenced documents.  
1203 In cases where a *CPP* or *CPA* document is in fact suitably signed, that facility could also  
1204 be used to ensure that the referenced *Process-Specification* documents are unchanged.  
1205 However, this specification does not currently mandate that a *CPP* or *CPA* be signed.

1206

1207 NOTE: If the *Parties* to a *CPA* wish to customize a previously existing *Process-*  
1208 *Specification* document, they MAY copy the existing document, modify it, and cause  
1209 their *CPA* to reference the modified copy. It is recognized that for reasons of clarity,  
1210 brevity, or historical record, the parties might prefer to reference a previously existing

1211        *Process-Specification* document in its original form and accompany that reference with a  
1212        specification of the agreed modifications. Therefore, *CPP* usage of the **ds:Reference**  
1213        element's **ds:Transforms** sub-element within a **ProcessSpecification** element might be  
1214        expanded in the future to allow other transforms as specified in the XML Digital  
1215        Signature specification[XMLDSIG]. For example, modifications to the original  
1216        document could then be expressed as XSLT transforms. After applying any transforms,  
1217        it would be necessary to validate the transformed document against the ebXML Business  
1218        Process Specification Schema[ebBPSS].  
1219

#### 1220        **8.4.5 Role element**

1221        The REQUIRED **Role** element identifies which role in the *Process Specification* the *Party* is  
1222        capable of supporting via the **ServiceBinding** element(s) siblings within this **CollaborationRole**  
1223        element.  
1224

1225        The **Role** element has the following attributes:

- 1226        • a REQUIRED **name** attribute,
- 1227        • a FIXED **xlink:type** attribute,
- 1228        • a REQUIRED **xlink:href** attribute.

##### 1229        **8.4.5.1 name attribute**

1230        The REQUIRED **name** attribute is a string that gives a name to the **Role**. Its value is taken from  
1231        one of the following sources in the *Process Specification*[ebBPSS] that is referenced by the  
1232        **ProcessSpecification** element depending upon which element is the "root" (highest order) of the  
1233        process referenced:  
1234

- 1235        • **name** attribute of a **BinaryCollaboration/initiatingRole** element,
- 1236        • **name** attribute of a **BinaryCollaboration/respondingRole** element,
- 1237        • **fromAuthorizedRole** attribute of a **BusinessTransactionActivity** element,
- 1238        • **toAuthorizedRole** attribute of a **BusinessTransactionActivity** element,
- 1239        • **fromAuthorizedRole** attribute of a **CollaborationActivity** element,
- 1240        • **toAuthorizedRole** attribute of a **CollaborationActivity** element,

1241        See NOTE in Section 8.4.4 regarding alternative *Business-Collaboration* descriptions.  
1242

##### 1243        **8.4.5.2 xlink:type attribute**

1244        The **xlink:type** attribute has a FIXED value of "simple". This identifies the element as being an  
1245        [XLINK] simple link.  
1246

##### 1247        **8.4.5.3 xlink:href attribute**

1248        The REQUIRED **xlink:href** attribute SHALL have a value that is a URI that conforms to  
1249        [RFC2396]. It identifies the location of the element or attribute within the *Process-Specification*  
1250        document that defines the role in the context of the *Business Collaboration*. An example is:  
1251

```
1252              xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"
```

1253        Where "Buyer" is the value of the ID attribute of the element in the *Process-Specification*  
1254        document that defines the role name.  
1255

1257

**1258 8.4.6 ApplicationCertificateRef element**

1259 The ***ApplicationCertificateRef*** element, if present, identifies a certificate for use by the business  
1260 process/application layer. This certificate is not used by the ebXML messaging system, but it is  
1261 included in the *CPP* so that it can be considered in the *CPA* negotiation process. The  
1262 ***ApplicationCertificateRef*** element can occur zero or more times.

1263 NOTE: It is up to the application software on both sides of a collaboration to determine  
1264 the intended/allowed usage of an application certificate by inspecting the key usage  
1265 extension within the certificate itself.

1266 NOTE: This element is included in the *CPP/CPA* to support interoperability with legacy  
1267 systems that already perform cryptographic functions such as digital signature or  
1268 encryption. Implementors should understand that use of ***ApplicationCertificateRef*** is  
1269 necessary only in cases where interoperability with such legacy systems is required.

1270

1271 The ***ApplicationCertificateRef*** element has

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

1272

**1273 8.4.6.1 certId attribute**

1274 The REQUIRED ***certId*** attribute is an [XML] IDREF that associates the ***CollaborationRole***  
1275 element with a certificate. It MUST have a value equal the value of the ***certId*** attribute of one of  
1276 the ***Certificate*** elements under ***PartyInfo***.

1277

**1278 8.4.7 ApplicationSecurityDetailsRef element**

1279 The ***ApplicationSecurityDetailsRef*** element, if present, identifies the trust anchors and security  
1280 policy that this *Party* will apply to any application-level certificate offered by the other *Party*.  
1281 These trust anchors and policy are not used by the ebXML messaging system, but are included in  
1282 the *CPP* so that they can be considered in the *CPA* negotiation process.

1283

1284 The ***ApplicationSecurityDetailsRef*** element has

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

1285

**1286 8.4.7.1 SecurityId attribute**

1287 The REQUIRED ***securityId*** attribute is an [XML] IDREF that associates the ***CollaborationRole***  
1288 with a ***SecurityDetails*** element that specifies a set of trust anchors and a security policy. It  
1289 MUST have a value equal to the value of the ***securityId*** attribute of one of the ***SecurityDetails***  
1290 elements under ***PartyInfo***.

1291

**1292 8.4.8 ServiceBinding element**

1293 The ***ServiceBinding*** element identifies a ***DeliveryChannel*** element for all of the business  
1294 *Message* traffic that is to be sent or received by the *Party* within the context of the identified  
1295 *Process-Specification* document. It MUST contain at least one ***CanReceive*** or ***CanSend*** child

1298 element.

1299  
1300 The **ServiceBinding** element has one child **Service** element, zero or more **CanSend** child  
1301 elements, and zero or more **CanReceive** child elements.  
1302

#### 1303 8.4.9 Service element

1304 The value of the **Service** element is a string that SHALL be used as the value of the **Service**  
1305 element in the ebXML *Message Header*[ebMS] or a similar element in the *Message Header* of  
1306 an alternative *message* service. The **Service** element has an IMPLIED **type** attribute.  
1307

1308 If the *Process-Specification* document is defined by the ebXML Business Process Specification  
1309 Schema[ebBPSS], then the value of the **Service** element MUST be the **uuid** (URI) attribute  
1310 specified for the **ProcessSpecification** element in the Business Process Specification Schema  
1311 instance document.  
1312

1313 NOTE: The purpose of the **Service** element is to provide routing information for the  
1314 ebXML *Message Header*. The **CollaborationRole** element and its child elements identify  
1315 the information in the **ProcessSpecification** document that is relevant to the *CPP* or *CPA*.  
1316 The **Service** element MAY be used along with the **CanSend** and **CanReceive** elements  
1317 (and their descendants) to provide routing of received messages to the correct application  
1318 entry point.  
1319

##### 1320 8.4.9.1 type attribute

1321 If the **type** attribute is present, it indicates that the *Parties* sending and receiving the *Message*  
1322 know, by some other means, how to interpret the value of the **Service** element. The two *Parties*  
1323 MAY use the value of the **type** attribute to assist the interpretation.  
1324

1325 If the **type** attribute is not present, the value of the **Service** element MUST be a URI[RFC2396].  
1326 If using the ebXML Business Process Specification[ebBPSS] for defining the *Process-*  
1327 *Specification* document, the type attribute MUST be a URI[RFC2396].  
1328

#### 1329 8.4.10 CanSend element

1330 The **CanSend** element identifies an *action* message that a *Party* is capable of sending. It has  
1331 three sub-elements: **ThisPartyActionBinding**, **OtherPartyActionBinding**, and **CanReceive**. The  
1332 **ThisPartyActionBinding** element is REQUIRED for both *CPPs* and *CPAs*. It identifies the  
1333 **DeliveryChannel** and the **Packaging** the *Party* described by the encompassing **PartyInfo**  
1334 element will use for sending the *action* invocation message in question. The  
1335 **OtherPartyActionBinding** element is only used in the case of *CPAs*. It is of type IDREF and  
1336 identifies a matching **ThisPartyActionBinding** element that is found under the collaboration  
1337 partner's **PartyInfo**. It indirectly identifies the **DeliveryChannel** the other *Party* will use for  
1338 receiving the *action* message in question and the expected **Packaging**. Within a *CPA* and under  
1339 the same **CanSend** element, the **DeliveryChannels** and **Packaging** used/expected by the two  
1340 *Parties* MUST be compatible. The **CanReceive** element can occur zero or more times. When  
1341 present, it indicates that one or more synchronous response actions are expected.  
1342 This is illustrated in the *CPP* and *CPA* examples in the appendices.

1343

#### 1344 8.4.11 CanReceive element

1345 The **CanReceive** element identifies an *action* invocation message that a *Party* is capable of  
1346 receiving. It has three sub-elements: **ThisPartyActionBinding**, **OtherPartyActionBinding**, and  
1347 **CanSend**. The **ThisPartyActionBinding** element is REQUIRED for both *CPPs* and *CPAs*. It  
1348 identifies the **DeliveryChannel** the *Party* described by the encompassing **PartyInfo** element will  
1349 use for receiving the *action* message in question and the **Packaging** it is expecting. The  
1350 **OtherPartyActionBinding** element is only used in the case of *CPAs*. It is of type IDREF and  
1351 identifies a matching **ThisPartyActionBinding** element that is found under the collaboration  
1352 partner's **PartyInfo**. It indirectly identifies the **DeliveryChannel** and **Packaging** the other *Party*  
1353 will use for sending the *action* invocation message in question. Within a *CPA* and under the  
1354 same **CanReceive** element, the **DeliveryChannels** and **Packaging** used/expected by the two  
1355 parties MUST be compatible. The **CanSend** element can occur zero or more times. When  
1356 present, it indicates that one or more synchronous response actions are expected. This is  
1357 illustrated in the *CPP* and *CPA* examples in the appendices.

1358

#### 1359 8.4.12 ThisPartyActionBinding element

1360 The **ThisPartyActionBinding** specifies one or more **DeliveryChannel** elements for *Messages* for  
1361 a selected *action* and the **Packaging** for those *Messages* that are to be sent or received by the  
1362 *Party* in the context of the *Process Specification* that is associated with the parent  
1363 **ServiceBinding** element.

1364

1365 The **ThisPartyActionBinding** element has a REQUIRED child  
1366 **BusinessTransactionCharacteristics** element, zero or one child **ActionContext** element and one  
1367 or more **ChannelID** child elements.

1368

1369 The **ThisPartyActionBinding** element has the following attributes:

- 1370 • a REQUIRED *action* attribute,
- 1371 • a REQUIRED *packageId* attribute,
- 1372 • an IMPLIED *xlink:href* attribute,
- 1373 • a FIXED *xlink:type* attribute.

1374

1375 Under a given **ServiceBinding** element, there MAY be multiple **CanSend** or **CanReceive** child  
1376 elements with the same *action* to allow different software entry points and Transport options. In  
1377 such a scenario, the **DeliveryChannels** referred by the **ChannelID** child elements of  
1378 **ThisPartyActionBinding** SHALL point to distinct **EndPoints** for the receiving MSH to uniquely  
1379 identify the **DeliveryChannel** being used for this particular message exchange.

1380

1381 NOTE: An implementation MAY provide the capability of dynamically assigning delivery  
1382 channels on a per *Message* basis during performance of the *Business Collaboration*. The delivery  
1383 channel selected would be chosen, based on present conditions, from those identified by  
1384 **CanSend** elements that refer to the *Business Collaboration* that is sending the *Message*. On the  
1385 receiving side, the MSH can use the distinct **EndPoints** to identify the **DeliveryChannel** used for  
1386 this message exchange.

1387  
1388 Within a **CanSend** element or a **CanReceive** element, when both the **ThisPartyActionBinding**  
1389 and **OtherPartyActionBinding** elements are present (i.e., in a *CPA*), they MUST have identical  
1390 action values or equivalent **ActionContext** elements. In addition, the **DeliveryChannel** and  
1391 **Packaging** that they reference MUST be compatible.

1392  
1393 **8.4.12.1 action attribute**  
1394 The value of the REQUIRED **action** attribute is a string that identifies the business document  
1395 exchange to be associated with the **DeliveryChannel** identified by the **ChannelId** sub-elements.  
1396 The value of the **action** attribute SHALL be used as the value of the **Action** element in the  
1397 ebXML *Message Header*[ebMS] or a similar element in the *Message Header* of an alternative  
1398 *message service*. The purpose of the **action** attribute is to provide a mapping between the  
1399 hierarchical naming associated with a *Business Process/Application* and the **Action** element in  
1400 the ebXML Message Header[ebMS]. This mapping MAY be implemented by using the  
1401 **ActionContext** element. See NOTE in Section 8.4.4 regarding alternative *Business*  
1402 *Collaboration* descriptions.

1403  
1404 Business signals, when sent individually (i.e., not bundled with response documents in  
1405 synchronous reply mode), SHALL use the values *ReceiptAcknowledgment*,  
1406 *AcceptanceAcknowledgment*, or *Exception* as the value of their **action** attribute. In addition, they  
1407 SHOULD specify a Service that is the same as the Service used for the original message.

1408  
1409 NOTE: In general, the action name chosen by the two parties to represent a particular  
1410 requesting business activity or responding business activity in the context of a binary  
1411 collaboration that makes use of nested binary collaborations MAY not be identical.  
1412 Therefore, when composing two *CPPs* to form a *CPA*, it is necessary to make use of  
1413 information from the associated **ActionContext** (see Section 8.4.15) in order to determine  
1414 if two different action names from the two *CPPs* actually represent the same  
1415 **ActionContext**. When business transactions are not reused in different contexts, it is  
1416 recommended that the names of the requesting business activity and responding business  
1417 activity be used as action names.

1418  
1419 **8.4.12.2 packageId attribute**  
1420 The REQUIRED **packageId** attribute is an [XML] IDREF that identifies the **Packaging** element  
1421 to be associated with the *Message* identified by the **action** attribute.

1422  
1423 **8.4.12.3 xlink:href attribute**  
1424 The IMPLIED **xlink:href** attribute, if present, SHALL provide an absolute [XPOINTER] URI  
1425 expression that specifically identifies the **RequestingBusinessActivity** or  
1426 **RespondingBusinessActivity** element within the associated *Process-Specification*  
1427 document[ebBPSS] that is identified by the **ProcessSpecification** element.

1428  
1429 **8.4.12.4 xlink:type attribute**  
1430 The IMPLIED **xlink:type** attribute has a FIXED value of "simple". This identifies the element as  
1431 being an [XLINK] simple link.

**1433 8.4.13 BusinessTransactionCharacteristics element**

1434 The **BusinessTransactionCharacteristics** element describes the security characteristics and other  
1435 attributes of the delivery channel, as derived from the **ProcessSpecification(s)** whose messages  
1436 are transported using the delivery channel. The attributes of the  
1437 **BusinessTransactionCharacteristics** element, MAY be used to override the values of the  
1438 corresponding attributes in the *Process-Specification* document.

1439  
1440 See NOTE in Section 8.4.4 regarding alternative *Business-Collaboration* descriptions.  
1441

1442 *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel  
1443 definitions for the sender and receiver (transport and document-exchange) for internal  
1444 consistency as well as compatibility between the two partners. Typically, when an attribute has a  
1445 particular value, sub-elements under the corresponding Transport and DocExchange elements  
1446 would exist to further describe the implied implementation parameters.  
1447

1448 The **BusinessTransactionCharacteristics** element has the following attributes:  
1449

- 1450 • an IMPLIED **isNonRepudiationRequired** attribute,
- 1451 • an IMPLIED **isNonRepudiationReceiptRequired** attribute,
- 1452 • an IMPLIED **isSecureTransportRequired** attribute,
- 1453 • an IMPLIED **isConfidential** attribute,
- 1454 • an IMPLIED **isAuthenticated** attribute,
- 1455 • an IMPLIED **isAuthorizationRequired** attribute,
- 1456 • an IMPLIED **isTamperProof** attribute,
- 1457 • an IMPLIED **isIntelligibleCheckRequired** attribute,
- 1458 • an IMPLIED **timeToAcknowledgeReceipt** attribute,
- 1459 • an IMPLIED **timeToAcknowledgeAcceptance** attribute,
- 1460 • an IMPLIED **timeToPerform** attribute,
- 1461 • an IMPLIED **retryCount** attribute.

1462 These attributes allow parameters specified at the *Process-Specification* level to be overridden. If  
1463 one of these attributes is not specified, the corresponding default value should be obtained from  
1464 the *Process-Specification* document.  
1465

**1466 8.4.13.1 isNonRepudiationRequired attribute**

1467 The **isNonRepudiationRequired** attribute is a Boolean with possible values of "true" and  
1468 "false". If the value is "true" then the delivery channel MUST specify that the *Message* is to be  
1469 digitally signed using the certificate of the *Party* sending the *Message*, and archived by both  
1470 parties. The **SenderNonRepudiation** element under **DocExchange/ebXMLSenderBinding** (see  
1471 Section 8.4.42) and the **ReceiverNonRepudiation** element under  
1472 **DocExchange/ebXMLReceiverBinding** (see Section 8.4.53) further describe various parameters  
1473 related to the implementation of non-repudiation of origin, such as the hashing algorithm, the  
1474 signature algorithm, the signing certificate, the trust anchor, etc.  
1475  
1476

**1477 8.4.13.2 isNonRepudiationReceiptRequired attribute**

1478 The ***isNonRepudiationReceiptRequired*** attribute is a Boolean with possible values of "true"  
1479 and "false". If the value is "true" then the delivery channel MUST specify that the *Message* is to  
1480 be acknowledged by a digitally signed *Receipt Acknowledgment* signal *Message*, signed using  
1481 the certificate of the *Party* that received the *Message*, that includes the digest(s) of the payload(s)  
1482 of the *Message* being acknowledged. The ***SenderNonRepudiation*** element under  
1483 ***DocExchange/ebXMLSenderBinding*** (see Section 8.4.42) and the ***ReceiverNonRepudiation***  
1484 element under ***DocExchange/ebXMLReceiverBinding*** (see Section 8.4.53) further describe  
1485 various parameters related to the implementation of non-repudiation of receipt.

**1487 8.4.13.3 isSecureTransportRequired attribute**

1488 The ***isSecureTransportRequired*** attribute is a Boolean with possible values of "true" and  
1489 "false". If the value is "true" then it indicates that the delivery channel uses a secure transport  
1490 protocol such as TLS[RFC2246] or [IPSEC]. Appendix E of the TLS Version 1.0  
1491 specification[RFC2246] covers backward compatibility with SSL [SSL].

**1493 8.4.13.4 isConfidential attribute**

1494 The ***isConfidential*** attribute has the possible values of "none", "transient", "persistent", and  
1495 "transient-and-persistent". These values MUST be interpreted as defined by the ebXML Business  
1496 Process Specification Schema[ebBPSS]. In general, transient confidentiality can be implemented  
1497 using a secure transport protocol like SSL; persistent confidentiality can be implemented using a  
1498 digital envelope mechanism like S/MIME. Secure transport information is further provided in the  
1499 ***TransportSender*** (see Section 8.4.24) and ***TransportReceiver*** (see Section 8.4.31) elements  
1500 under the ***Transport*** element. Persistent encryption information is further provided in the  
1501 ***SenderDigitalEnvelope*** element under ***DocExchange/ebXMLSenderBinding*** (see Section  
1502 8.4.47) and the ***ReceiverDigitalEnvelope*** element under ***DocExchange/ebXMLReceiverBinding***  
1503 (see Section 8.4.55).

1504 The *CPA* would be inconsistent if ***isConfidential*** is set to "transient" or "persistent-and-  
1505 transient", while ***isSecureTransportRequired*** is set to "false".

**1508 8.4.13.5 isAuthenticated attribute**

1509 The ***isAuthenticated*** attribute has the possible values of "none", "transient", "persistent", and  
1510 "persistent-and-transient". If this attribute is set to any value other than "none", then the receiver  
1511 MUST be able to verify the identity of the sender. In general, transient authentication can be  
1512 implemented using a secure transport protocol like SSL (with or without the use of basic or  
1513 digest authentication); persistent authentication can be implemented using a digital signature  
1514 mechanism. Secure transport information is further provided in the ***TransportSender*** (see  
1515 Section 8.4.24) and ***TransportReceiver*** (see Section 8.4.32) elements under the ***Transport***  
1516 element. Persistent authentication information is further provided in the ***SenderNonRepudiation***  
1517 element under ***DocExchange/ebXMLSenderBinding*** (see Section 8.4.42) and the  
1518 ***ReceiverNonRepudiation*** element (under ***DocExchange/ebXMLReceiverBinding*** (see Section  
1519 8.4.53)).

1520 The *CPA* would be inconsistent if ***isAuthenticated*** is set to "transient" or "persistent-and-  
1521 transient", while ***isSecureTransportRequired*** is set to "false".

1523

**8.4.13.6 isAuthorizationRequired attribute**

1524 The ***isAuthorizationRequired*** attribute is a Boolean with possible values of "true" and  
1525 "false". If the value is "true" then it indicates that the delivery channel MUST specify that the  
1526 sender of the *Message* is to be authorized before delivery to the application.

1527

**8.4.13.7 isTamperProof attribute**

1528 The ***isTamperProof*** attribute has the possible values of "none", "transient", "persistent", and  
1529 "persistent-and-transient". If this attribute is set to a value other than "none", then it must be  
1530 possible for the receiver to detect if the received message has been corrupted or tampered with.  
1531 In general, transient tamper detection can be implemented using a secure transport like SSL;  
1532 persistent tamper detection can be implemented using a digital signature mechanism. Secure  
1533 transport information is further provided in the ***TransportSender*** (see Section 8.4.24) and  
1534 ***TransportReceiver*** (see Section 8.4.47) elements under the ***Transport*** element. Digital signature  
1535 information is further provided in the ***SenderNonRepudiation*** element under  
1536 ***DocExchange/ebXMLSenderBinding*** (see Section 8.4.42) and the ***ReceiverNonRepudiation***  
1537 element under ***DocExchange/ebXMLReceiverBinding*** (see Section 8.4.53).

1538

1539 The *CPA* would be inconsistent if ***isTamperProof*** is set to "transient" or "persistent-and-  
1540 transient", while ***isSecureTransportRequired*** is set to "false".

1541

**8.4.13.8 isIntelligibleCheckRequired attribute**

1542 The ***isIntelligibleCheckRequired*** attribute is a Boolean with possible values of "true" and  
1543 "false". If the value is "true", then the receiver MUST verify that a business document is not  
1544 garbled (i.e., passes schema validation) before returning a *Receipt Acknowledgment* signal.

1545

**8.4.13.9 timeToAcknowledgeReceipt attribute**

1546 The ***timeToAcknowledgeReceipt*** attribute is of type duration [XMLSCHEMA-2]. It specifies the  
1547 time period within which the receiving *Party* has to acknowledge receipt of a business document.

1548

1549 If this attribute is specified, then the *Receipt Acknowledgment* signal MUST be used.

1550

**8.4.13.10 timeToAcknowledgeAcceptance attribute**

1551 The ***timeToAcknowledgeAcceptance*** attribute is of type duration [XMLSCHEMA-2]. It  
1552 specifies the time period within which the receiving *Party* has to non-substantively acknowledge  
1553 acceptance of a business document (i.e., after it has passed business rules validation).

1554 If this attribute is specified, then the *Acceptance Acknowledgment* signal MUST be used.

1555

**8.4.13.11 timeToPerform attribute**

1556 The ***timeToPerform*** attribute is of type duration [XMLSCHEMA-2]. It specifies the time period,  
1557 starting from the initiation of the ***RequestingBusinessActivity***, within which the initiator of the  
1558 transaction MUST have received the response, i.e., the business document associated with the  
1559 ***RespondingBusinessActivity***.

1560

1561 NOTE: The ***timeToPerform*** attribute associated with a ***BinaryCollaboration*** in BPSS is  
1562 currently not modeled in this specification. Therefore, it cannot be overridden. In other

1569 words, the value specified at the BPSS level MUST be used.  
1570

1571 When synchronous reply mode is in use (see Section 8.4.22.1), the **TimeToPerform** value  
1572 SHOULD be used as the connection timeout.

#### 1573 8.4.13.12 retryCount attribute

1574 The **retryCount** attribute is of type integer. It specifies the maximum number of times the  
1575 *Business Transaction* MAY be retried should certain error conditions (e.g., time out waiting for  
1576 the Receipt Acknowledgment signal) arise during its execution. Such retries MUST not be used  
1577 when ebXML Reliable Messaging is employed to transport messages in the *Business*  
1578 *Transaction*. In the latter case, retries are governed by the **Retry**, **RetryInterval** elements under  
1579 the **ReliableMessaging** element.

#### 1581 8.4.14 ChannelId element

1582 The **ChannelId** element identifies one or more **DeliveryChannel** elements that can be used for  
1583 sending or receiving the corresponding action messages. Multiple **ChannelId** elements can be  
1584 used to associate **DeliveryChannel** elements with different characteristics with the same  
1585 **CanSend** or **CanReceive** element. For example, a *Party* that supports both HTTP and SMTP for  
1586 sending the same action can specify different **ChannelId** attribute values for the corresponding  
1587 channels. If using multiple **DeliveryChannel** elements, different **EndPoint** elements MUST be  
1588 used, so that the receiving MSH can uniquely determine the **DeliveryChannel** element being  
1589 used for this message exchange.

#### 1591 8.4.15 ActionContext element

1592 The **ActionContext** element provides a mapping from the **action** attribute in the  
1593 **ThisPartyActionBinding** element to the corresponding *Business Process* implementation-  
1594 specific naming strategy, if any. If the *Process-Specification* document is defined by the ebXML  
1595 Business Process Specification Schema[ebBPSS], the **ActionContext** element MUST be present.

1596 Any business process/application implementation can use a combination of information in the  
1597 **action** attribute and the **ActionContext** elements to make message routing decisions. If using  
1598 alternative *Business-Collaboration* description schemas, the **action** attribute of the parent  
1599 **ThisPartyActionBinding** element and/or the [XMLSCHEMA-1] **wildcard** element within the  
1600 **ActionContext** element MAY be used to make routing decisions above the level of the *Message*  
1601 Service Handler.

1602 The **ActionContext** element has the following elements:

- 1603 • zero or one **CollaborationActivity** element,
- 1604 • zero or more [XML SCHEMA-1] **wildcard** elements.

1605 The **ActionContext** element also has the following attributes:

- 1606 • a REQUIRED **binaryCollaboration** attribute,
- 1607 • a REQUIRED **businessTransactionActivity** attribute,
- 1608 • a REQUIRED **requestOrResponseAction** attribute.

1613

#### 1614 8.4.15.1 **binaryCollaboration** attribute

1615 The REQUIRED **binaryCollaboration** attribute is a string that identifies the  
1616 **BinaryCollaboration** for which the parent **ThisPartyActionBinding** is defined. If the *Process-*  
1617 *Specification* document is defined by the ebXML Business Process Specification  
1618 Schema[ebBPSS], then the value of the **binaryCollaboration** attribute MUST match the value of  
1619 the **name** attribute of the **BinaryCollaboration** element as defined in the ebXML Business  
1620 Process Specification Schema[ebBPSS].

1621

#### 1622 8.4.15.2 **businessTransactionActivity** attribute

1623 The REQUIRED **businessTransactionActivity** attribute is a string that identifies the *Business*  
1624 *Transaction* for which the parent **ThisPartyActionBinding** is defined. If the *Process-*  
1625 *Specification* document is defined by the ebXML Business Process Specification  
1626 Schema[ebBPSS], the value of the **businessTransactionActivity** attribute MUST match the value  
1627 of the **name** attribute of the **BusinessTransactionActivity** element, whose parent is the *Binary*  
1628 *Collaboration* referred to by the **binaryCollaboration** attribute.

1629

#### 1630 8.4.15.3 **requestOrResponseAction** attribute

1631 The REQUIRED **requestOrResponseAction** attribute is a string that identifies either the  
1632 *Requesting or Responding Business Activity* for which the parent **ThisPartyActionBinding** is  
1633 defined. For a **ThisPartyActionBinding** defined for the request side of a message exchange, if  
1634 the *Process-Specification* document is defined by the ebXML Business Process Specification  
1635 Schema [ebBPSS], the value of the **requestOrResponseAction** attribute MUST match the value  
1636 of the **name** attribute of the **RequestingBusinessActivity** element corresponding to the *Business*  
1637 *Transaction* specified in the **businessTransactionActivity** attribute. Similarly, for the response  
1638 side of a message exchange, the value of the **requestOrResponseAction** attribute MUST match  
1639 the value of the **name** attribute of the **RespondingBusinessActivity** element corresponding to the  
1640 *Business Transaction* specified in the **businessTransactionActivity** attribute, as defined in the  
1641 ebXML Business Process Specification Schema[ebBPSS].

1642

### 1643 8.4.16 **CollaborationActivity** element

1644 The **CollaborationActivity** element supports the ActionContext element by providing the ability  
1645 to map any nested *Binary Collaborations* as defined in the ebXML Business Process  
1646 Specification Schema[ebBPSS] to the **action** attribute. The **CollaborationActivity** element  
1647 MUST be present when the *Binary Collaboration* referred to by the **binaryCollaboration**  
1648 attribute has a *Collaboration Activity* defined in the business process definition.

1649

1650 An example of the **CollaborationActivity** element is:

1651  
1652       <tp:CollaborationActivity  
1653            tp:name="Credit Check" />  
1654

1655 The **CollaborationActivity** element has zero or one child **CollaborationActivity** element to  
1656 indicate further nesting of *Binary Collaborations*.

1657

1658 The **CollaborationActivity** element also has one attribute:

- 1659           • a REQUIRED **name** attribute.  
1660

1661     **8.4.16.1 name attribute**

1662     The REQUIRED **name** attribute is a string that identifies the *Collaboration Activity* included in  
1663     the *Binary Collaboration*. If the *Process-Specification* document is defined by the ebXML  
1664     Business Process Specification Schema[ebBPSS], the value of the **name** attribute MUST match  
1665     the value of the **name** attribute of the *CollaborationActivity* within the *BinaryCollaboration*, as  
1666     defined in the ebXML Business Process Specification Schema[ebBPSS].  
1667

1668     **8.4.17 Certificate element**

1669     The **Certificate** element defines certificate information for use in this *CPP*. One or more  
1670     **Certificate** elements can be provided for use in the various security functions in the *CPP*. An  
1671     example of the **Certificate** element is:

1672       <tp:Certificate tp:certId="CompanyA\_SigningCert">  
1673         <ds:KeyInfo> . . . </ds:KeyInfo>  
1674       </tp:Certificate>  
1675

1676     The **Certificate** element has a single REQUIRED attribute: **certId**. The **Certificate** element has a  
1677     single child element: **ds:KeyInfo**.

1678     The **ds:KeyInfo** element may contain a complete chain of certificates, but the leaf certificate is  
1679     the **Certificate** element containing the key used in various asymmetric cryptographic operations.  
1680     (The leaf certificate will be one that has been issued but has not been used to issue certificates.)  
1681     If the leaf certificate has been issued by an intermediate certificate authority, the complete chain  
1682     to the root certificate authority SHOULD be included because it aids in testing certificate  
1683     validity with respect to a set of trust anchors.  
1684

1685     **8.4.17.1 certId attribute**

1686     The REQUIRED **certId** attribute is an [XML] ID that is referred to by a **CertificateRef** element  
1687     elsewhere in the *CPP*. Here is an example of how a **CertificateRef** would refer to the **Certificate**  
1688     element shown in the previous section:  
1689

1690       <tp:SigningCertificateRef tp:certId="CompanyA\_SigningCert" />  
1691

1692     **8.4.17.2 ds:KeyInfo element**

1693     The **ds:KeyInfo** element defines the certificate information. The content of this element and any  
1694     sub-elements are defined by the XML Digital Signature specification[XMLDSIG].  
1695

1696       NOTE: Software for creation of *CPPs* and *CPAs* MUST recognize the **ds:KeyInfo**  
1697       element and insert the sub-element structure necessary to define the certificate.  
1698

1699     **8.4.18 SecurityDetails element**

1700     The **SecurityDetails** element defines a set of **TrustAnchors** and an associated **SecurityPolicy** for  
1701     use in this *CPP*. One or more **SecurityDetails** elements can be provided for use in the various  
1702

1704 security functions in the *CPP*. An example of the **SecurityDetails** element is:

```
1705      <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
1706          <tp:TrustAnchors tp:trustId="MessageTrustAnchors">
1707              <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
1708              <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
1709          </tp:TrustAnchors>
1710          <tp:SecurityPolicy> ... </tp:SecurityPolicy>
1711      </tp:SecurityDetails>
```

1713  
1714 The **SecurityDetails** element has zero or one **TrustAnchors** element that identify a set of  
1715 certificates that are trusted by the *Party*. It also has zero or one **SecurityPolicy** element.

1716  
1717 The **SecurityDetails** element allows agreement to be reached on what root certificates will be  
1718 used in checking the validity of the other *Party*'s certificates. It can also specify policy regarding  
1719 operation of the public key infrastructure.

1720  
1721 The **SecurityDetails** element has one attribute:  
1722 • A REQUIRED **securityId** attribute.

#### 1723 8.4.18.1 **securityId** attribute

1724 The REQUIRED **securityId** attribute is an [XML] ID that is referred to by an element elsewhere  
1725 in the *CPP*. Here is an example of how a **SigningSecurityDetailsRef** would refer to the  
1726 **SecurityDetails** element shown in the previous section:

```
1727      <tp:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity" />
```

#### 1731 8.4.19 **TrustAnchors** element

1732 The **TrustAnchors** element contains one or more **AnchorCertificateRef** elements, each of which  
1733 refers to a **Certificate** element (under **PartyInfo**) that represents a certificate trusted by this  
1734 *Party*. These trusted certificates are used in the process of certificate path validation. If a  
1735 certificate in question does not "chain" to one of this *Party*'s trust anchors, it is considered  
1736 invalid.

1737  
1738 The TrustAnchors element eventually resolves into XMLDSig **KeyInfo** elements. These elements  
1739 may contain several certificates (a chain), and may refer to those certificates using the  
1740 **RetrievalMethod** element. When there is a chain, the trust anchor is the "leaf" certificate with  
1741 respect to the "root" issuing certificate authority (CA) certificate. The root CA will be a self-  
1742 issued and self-signed certificate, and using the Issuer information and perhaps key usage  
1743 attributes, the leaf certificate ("issued but not issuing" within the chain) can be determined. The  
1744 chain is included for convenience in that validity checks typically will chain to a "root" CA.  
1745 Please note that the inclusion of a root CA in a chain does not mean that the root CA is being  
1746 announced as a trust anchor. It is possible for there to be a PKI policy in which some, but not all,  
1747 intermediate CAs are trusted. If a root CA were accepted as a trust anchor, all of its intermediate  
1748 CAs, and all the certificates they issue, would be validated. That might not be what was intended.

#### 1750 8.4.20 SecurityPolicy element

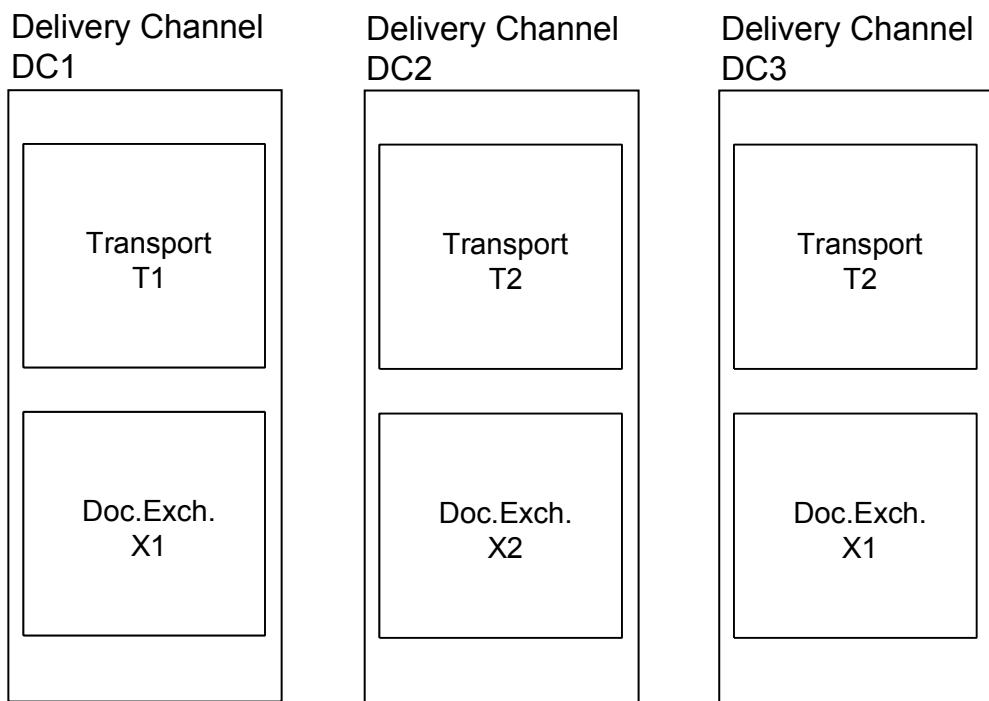
1751 The **SecurityPolicy** element is a placeholder for future apparatus that will enable the *Party* to  
1752 specify its policy and compliance regarding specific components of its public key infrastructure.  
1753 For example, it might stipulate revocation checking procedures or constraints related to name,  
1754 usage, or path length.

1755

#### 1756 8.4.21 DeliveryChannel element

1757 A delivery channel is a combination of a **Transport** element and a **DocExchange** element that  
1758 describes the *Party's Message* communication characteristics. The *CPP* SHALL contain one or  
1759 more **DeliveryChannel** elements, one or more **Transport** elements, and one or more  
1760 **DocExchange** elements. Each delivery channel SHALL refer to any combination of a  
1761 **DocExchange** element and a **Transport** element. The same **DocExchange** element or the same  
1762 **Transport** element can be referred to by more than one delivery channel. Two delivery  
1763 channels can use the same transport protocol and the same document-exchange protocol and  
1764 differ only in details such as communication addresses or security definitions. Figure 5 illustrates  
1765 three delivery channels.

**Figure 5: Three Delivery Channels**



1766

1767 The delivery channels have ID attributes with values "DC1", "DC2", and "DC3". Each delivery  
1768 channel contains one transport definition and one document-exchange definition. Each transport  
1769 definition and each document-exchange definition also has an ID attribute whose value is shown  
1770 in the figure. Note that delivery channel DC3 illustrates that a delivery channel can refer to the  
1771 same transport definition and document-exchange definition used by other delivery channels but  
1772 a different combination. In this case delivery channel DC3 is a combination of transport

1773 definition T2 (also referred to by delivery channel DC2) and document-exchange definition X1  
1774 (also referred to by delivery channel DC1).

1775

1776 Following is the delivery-channel syntax.

1777

```
<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>
```

1789

1790 Each **DeliveryChannel** element identifies one **Transport** element and one **DocExchange** element  
1791 that together make up a single delivery channel definition.

1792

1793 The **DeliveryChannel** element has the following attributes:

1794

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

1797

1798 The **DeliveryChannel** element has one REQUIRED child element, **MessagingCharacteristics**.

1799

#### 8.4.21.1 channelId attribute

1800

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*.

1803

#### 8.4.21.2 transportId attribute

1804

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.

1808

#### 8.4.21.3 docExchangeId attribute

1809

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.

1814

### 8.4.22 MessagingCharacteristics element

1816

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.

1820

1821 *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel  
**Collaboration-Protocol Profile and Agreement Specification**

1822 definition (transport and document-exchange) for consistency with these attributes.  
1823

1824 The ***MessagingCharacteristics*** element has the following attributes:

- 1825     • An IMPLIED ***syncReplyMode*** attribute,  
1826     • an IMPLIED ***ackRequested*** attribute,  
1827     • an IMPLIED ***ackSignatureRequested*** attribute,  
1828     • an IMPLIED ***duplicateElimination*** attribute,  
1829     • an IMPLIED ***actor*** attribute.

1830

#### 1831 8.4.22.1 syncReplyMode attribute

1832 The ***syncReplyMode*** attribute is an enumeration comprised of the following possible values:

- 1833     • "mshSignalsOnly"  
1834     • "signalsOnly"  
1835     • "responseOnly"  
1836     • "signalsAndResponse"  
1837     • "none"

1838  
1839 This attribute, when present, indicates what the sending application expects in a synchronous  
1840 response (the delivery channel MUST be bound to a synchronous communication protocol such  
1841 as HTTP when ***syncReplyMode*** is not "none").

1842  
1843 The value of "mshSignalsOnly" indicates that the response returned (on the HTTP 200 response  
1844 in the case of HTTP) will only contain standalone *Message Service Handler (MSH)* level  
1845 messages like Acknowledgment (for Reliable Messaging) and Error messages. All other  
1846 application level responses are to be returned asynchronously (using a ***DeliveryChannel*** element  
1847 determined by the service and action in question).

1848  
1849 The value of "signalsOnly" indicates that the response returned (on the HTTP 200 response in  
1850 the case of HTTP) will only include one or more *Business* signals as defined in the *Process-*  
1851 *Specification* document[ebBPSS], plus any piggybacked MSH level signals, but not a *Business-*  
1852 *response Message*. If the *Process-Specification* calls for the use of a *Business-response Message*,  
1853 then the latter MUST be returned asynchronously. If the *Business Process* does not call for the  
1854 use of an *Acceptance Acknowledgment* signal, then the ***Action*** element in the synchronously  
1855 returned ebXML *Message* MUST be set to "ReceiptAcknowledgment". Otherwise, the ***Action***  
1856 element in the synchronously returned ebXML *Message* (which includes both a *Receipt*  
1857 *Acknowledgment* signal and an *Acceptance Acknowledgment* signal) MUST be set to  
1858 "AcceptanceAcknowledgment".

1859  
1860 The value of "responseOnly" indicates that any *Business* signals, even if they are indicated in the  
1861 *Process Specification*, are to be omitted and only the *Business-response Message* will be returned  
1862 synchronously, plus any piggybacked MSH level signals. To be consistent, the  
1863 ***timeToAcknowledgeReceipt*** and ***timeToAcknowledgeAcceptance*** attributes under the  
1864 corresponding ***BusinessTransactionCharacteristics*** element SHOULD be set to zero to indicate  
1865 that these signals are not to be used at all. The ***Action*** element in the synchronously returned  
1866 ebXML *Message* is determined by the name of the action in the *CPA* that corresponds to the  
1867 appropriate ***RespondingBusinessActivity*** in the *Business Process*.

1868  
1869 The value of "signalsAndResponse" indicates that the application will synchronously return the  
1870 *Business-response Message* in addition to one or more *Business* signals, plus any piggybacked  
1871 *MSH* level signals. In this case, each signal and response that is bundled into the same ebXML  
1872 message must appear as a separate MIME part (i.e., be placed in a separate payload container).  
1873 To be consistent, the *timeToAcknowledgeReceipt* and *timeToPerform* attributes under the  
1874 corresponding **BusinessTransactionCharacteristics** element SHOULD have identical values.  
1875 The *timeToAcknowledgeAcceptance* attribute, if specified, SHOULD also have the same value  
1876 as the above two timing attributes. The **Action** element in the synchronously returned ebXML  
1877 *Message* is determined by the name of the action in the *CPA* that corresponds to the appropriate  
1878 **RespondingBusinessActivity** in the *Business Process*.  
1879  
1880 The *Receipt Acknowledgment* signal for the *Business-response Message*, sent from the request  
1881 initiator back to the responder, if called for by the *Process-Specification*, MUST also be  
1882 delivered over the same synchronous connection.  
1883  
1884 NOTE: For HTTP 1.1 clients and servers, two HTTP requests and replies will have to be  
1885 sent and received on the same connection. Implementations that implicitly assume that a  
1886 HTTP connection will be closed after a single synchronous request reply interchange will  
1887 not be able to support the "signalsAndResponse" synchronous reply mode.  
1888  
1889 The value of "none", which is the implied default value in the absence of the **syncReplyMode**  
1890 attribute, indicates that neither the *Business-response Message* nor any *Business* signal(s) will be  
1891 returned synchronously. In this case, all *Message Service Handler* level and *Business* level  
1892 messages will be returned as separate asynchronous messages.  
1893  
1894 The ebXML *Message Service*'s **SyncReply** element is included in the SOAP Header whenever  
1895 the **syncReplyMode** attribute has a value other than "none". If the delivery channel identifies a  
1896 transport protocol that has no synchronous capabilities (such as SMTP), the  
1897 **BusinessTransactionCharacteristics** element SHALL NOT have a **syncReplyMode** attribute  
1898 with a value other than "none".  
1899  
1900 When the value of the **syncReplyMode** attribute is other than "none", a synchronous delivery  
1901 channel SHALL be used to exchange all messages necessary for conducting a business  
1902 transaction. If the *Process Specification* calls for the use of non-repudiation of receipt for the  
1903 response message, then the initiator is expected to return a signed *ReceiptAcknowledgment* signal  
1904 for the responder's response message.  
1905  
1906 **8.4.22.2 ackRequested attribute**  
1907 The IMPLIED **ackRequested** attribute is an enumeration comprised of the following possible  
1908 values:  
1909     

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

  
1910  
1911  
1912  
1913 This attribute has the default value "perMessage" meaning whether the **AckRequested** element in

1914 the SOAP Header is present or absent can be varied on a "per message" basis. If this attribute is  
1915 set to "always", then every message sent over the delivery channel MUST have an *AckRequested*  
1916 element in the SOAP Header. If this attribute is set to "never", then every message sent over the  
1917 delivery channel MUST NOT have an *AckRequested* element in the SOAP Header.

1918  
1919 If the *ackRequested* attribute is not set to "never", then the *ReliableMessaging* element must be  
1920 present under the corresponding *DocExchange* element to provide the necessary Reliable  
1921 Messaging parameters.

#### 1923 **8.4.22.3 ackSignatureRequested attribute**

1924 The IMPLIED *ackSignatureRequested* attribute is an enumeration comprised of the following  
1925 possible values:

- 1926 • "always"
- 1927 • "never"
- 1928 • "perMessage"

1930 This attribute determines how the *signed* attribute within the *AckRequested* element in the SOAP  
1931 Header is to be set. It has the default value "perMessage" meaning that the *signed* attribute in the  
1932 *AckRequested* element within the SOAP Header can be set to "true" or "false" on a "per  
1933 message" basis. If this attribute is set to "always", then every message sent over the delivery  
1934 channel that has an *AckRequested* element in the SOAP Header MUST have its *signed* attribute  
1935 set to "true". If this attribute is set to "never", then every message sent over the delivery channel  
1936 that has an *AckRequested* element in the SOAP Header MUST have its *signed* attribute set to  
1937 "false". If the *ackRequested* attribute is set to "never", the setting of the *ackSignatureRequested*  
1938 attribute has no effect.

1939  
1940 NOTE: By enabling the use of signed *Acknowledgment* for reliably delivered messages,  
1941 a weak form of non-repudiation of receipt can be supported. This is considered weaker  
1942 than the *Receipt Acknowledgment* signal because no schema check can be performed on  
1943 the payload prior to the return of the *Acknowledgment*. The *ackSignatureRequested*  
1944 attribute can be set independent of the value for the *isNonRepudiationReceiptRequired*  
1945 attribute under the *Business Transaction Characteristics* element. Thus, even if the  
1946 original *Process-Specification* specifies that non-repudiation of receipt is to be  
1947 performed, the *CPP* and/or *CPA* can override this requirement, set  
1948 *isNonRepudiationReceiptRequired* to "false" and *ackSignatureRequested* to "always"  
1949 and thereby achieve the weak form of non-repudiation of receipt.

#### 1951 **8.4.22.4 duplicateElimination attribute**

1952 The IMPLIED *duplicateElimination* attribute is an enumeration comprised of the following  
1953 possible values:

- 1954 • "always"
- 1955 • "never"
- 1956 • "perMessage"

1957  
1958 This attribute determines whether the *DuplicateElimination* element within the *MessageHeader*  
1959 element in the SOAP Header is to be present. It has the default value "perMessage" meaning that

1960 the **DuplicateElimination** element within the SOAP Header can be present or absent on a "per  
1961 message" basis. If this attribute is set to "always", then every message sent over the delivery  
1962 channel MUST have a **DuplicateElimination** element in the SOAP Header. If this attribute is set  
1963 to "never", then every message sent over the delivery channel MUST NOT have a  
1964 **DuplicateElimination** element in the SOAP Header. If the **duplicateElimination** attribute is not  
1965 set to "never", then the **PersistDuration** element must be present under the corresponding  
1966 **DocExchange** element to provide the necessary persistent storage parameter.  
1967

#### 1968 8.4.22.5 actor attribute

1969 The IMPLIED **actor** attribute is an enumeration of the following possible values:

- 1970 • "urn:oasis:names:tc:ebxml-msg:actor:nextMSH"
- 1971 • "urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"

1972  
1973 This is a URI that will be used as the value for the **actor** attribute in the **AckRequested** element  
1974 (see [ebMS]) in case the latter is present in the SOAP Header, as governed by the **ackRequested**  
1975 attribute within the **MessagingCharacteristics** element in the **CPA**. If the **ackRequested** attribute  
1976 is set to "never", the setting of the **actor** attribute has no effect.  
1977

#### 1978 8.4.23 Transport element

1979 The **Transport** element defines the *Party's* network communication capabilities. One or more  
1980 **Transport** elements MUST be present in a **CPP**, each of which describes a mechanism the *Party*  
1981 uses to send messages, a mechanism it uses to receive messages, or both. The following example  
1982 illustrates the structure of a typical **Transport** element:  
1983

```
1984 <tp:Transport tp:transportId="transportA1">
1985   <tp:TransportSender> <!-- 0 or 1 time -->
1986     <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
1987     <tp:TransportClientSecurity>
1988       <tp:TransportSecurityProtocol tp:version="3.0">
1989         SSL
1990       </tp:TransportSecurityProtocol>
1991       <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert" />
1992       <tp:ServerSecurityDetailsRef
1993         tp:securityId="CompanyA_TransportSecurity" />
1994     </tp:TransportClientSecurity>
1995   </tp:TransportSender>
1996   <tp:TransportReceiver> <!-- 0 or 1 time -->
1997     <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
1998     <tp:Endpoint
1999       tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler"
2000       tp:type="allPurpose" />
2001     <tp:TransportServerSecurity>
2002       <tp:TransportSecurityProtocol tp:version="3.0">
2003         SSL
2004       </tp:TransportSecurityProtocol>
2005       <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert" />
2006       <tp:ClientSecurityDetailsRef
2007         tp:securityId="CompanyA_TransportSecurity" />
2008     </tp:TransportServerSecurity>
2009   </tp:TransportReceiver>
2010 </tp:Transport>
```

2011  
2012 The **Transport** element consists of zero or one **TransportSender** element and zero or one

2013     **TransportReceiver** element.

2014

2015     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  
2016     communicate in synchronous mode (where replies are returned over the same TCP connections  
2017     messages are sent on; see Section 8.4.22.1), its *CPP* MUST provide a **ServiceBinding** that  
2018     contains **ActionBindings** that are bound to a **DeliveryChannel** that uses a bi-directional  
2019     **Transport**.

2020

2021

2022

2023     A **Transport** that contains either a **TransportSender** or a **TransportReceiver** element, but not  
2024     both, is said to be *unidirectional*. A unidirectional **Transport** can only be used for sending or  
2025     receiving messages (not both) depending on which element it includes.

2026

2027     A *CPP* contains as many **Transport** elements as are needed to fully express the *Party*'s inbound  
2028     and outbound communication capabilities. If, for example, the *Party* can send and receive  
2029     messages via HTTP and SMTP, its *CPP* would contain a **Transport** element containing its HTTP  
2030     properties and another **Transport** element containing its SMTP properties.

2031

2032     The **Transport** element has

- a REQUIRED **transportId** attribute.

2033

#### 2034     **8.4.23.1 transportId attribute**

2035     The REQUIRED **transportId** attribute is an [XML] ID that refers to a **Transport** element  
2036     elsewhere in the *CPP*. Here is an example of a **DeliveryChannel** that refers to the **Transport**  
2037     element shown in the previous section:

2038

```
2039      <tp:DeliveryChannel tp:channelId="channelA1"  
2040          tp:transportId="transportA1"  
2041          tp:docExchangeId="docExchangeA1">  
2042          <tp:MessagingCharacteristics . . . />  
2043      </tp:DeliveryChannel>
```

2044

2045

#### 2046     **8.4.24 TransportSender element**

2047     The **TransportSender** element contains properties related to the sending side of a  
2048     **DeliveryChannel**. Its REQUIRED **TransportProtocol** element specifies the transport protocol  
2049     that will be used for sending messages. The **AccessAuthentication** element(s), if present,  
2050     specifies the type(s) of access authentication supported by the client. The  
2051     **TransportClientSecurity** element, if present, defines the *Party*'s provisions for client-side  
2052     transport layer security.

2053

2054

2055     The **TransportSender** element has no attributes.

2056

#### 2057     **8.4.25 TransportProtocol element**

2058     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

2059 of the protocol.

2060  
2061  
2062  
2063

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

2064 **8.4.26 AccessAuthentication element**

2065 The **AccessAuthentication** element, if present, indicates the authentication mechanism that MAY  
2066 be used by a transport server to challenge a client request and by a client to provide  
2067 authentication information to a server. For example, [RFC2617] specifies two access  
2068 authentication schemes for HTTP: "basic" and "digest". A client that supports both would have  
2069 two **AccessAuthentication** elements, as shown below. When multiple schemes are supported, the  
2070 order in which they are specified in the CPP indicates the order of preference.  
2071

```
<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>
```

2081 NOTE: A **CPA** will contain, for each **TransportSender** or **TransportReceiver**, only the  
2082 agreed-upon **AccessAuthentication** elements.

2083  
2084  
2085  
2086

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

2087 **8.4.27 TransportClientSecurity element**

2088 The **TransportClientSecurity** element provides information about this *Party*'s transport client  
2089 needed by the other *Party*'s transport server to enable a secure connection to be established  
2090 between the two. It contains a REQUIRED **TransportSecurityProtocol** element, zero or one  
2091 **ClientCertificateRef** element, zero or one **ServerSecurityDetailsRef** element, and zero or more  
2092 **EncryptionAlgorithm** elements.

2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100

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.

2101 **8.4.28 TransportSecurityProtocol element**

2102 The **TransportSecurityProtocol** element identifies the transport layer security protocol that is  
2103 supported by the parent **Transport**. The IMPLIED **version** attribute identifies the specific version  
2104 of the protocol.  
2105

2106 For encryption, the protocol is TLS Version 1.0[RFC2246], which uses public-key encryption.  
2107 Appendix E of the TLS Version 1.0 specification[RFC2246] covers backward compatibility with  
2108 SSL [SSL].  
2109

#### 2110 **8.4.29 ClientCertificateRef element**

2111 The ***ClientCertificateRef*** element identifies the certificate to be used by the client's transport  
2112 security module. The REQUIRED IDREF attribute ***certId*** identifies the certificate to be used by  
2113 referring to the ***Certificate*** element (under ***PartyInfo***) that has the matching ID attribute value. A  
2114 TLS-capable HTTP client, for example, uses this certificate to authenticate itself with receiver's  
2115 secure HTTP server.  
2116

2117 The ***ClientCertificateRef*** element, if present, indicates that mutual authentication between client  
2118 and server (i.e., initiator and responder of the HTTP connection) MUST be performed.  
2119

2120 The ***ClientCertificateRef*** element has

- 2121 • A REQUIRED ***certId*** attribute.  
2122

#### 2123 **8.4.30 ServerSecurityDetailsRef element**

2124 The ***ServerSecurityDetailsRef*** element identifies the trust anchors and security policy that this  
2125 *Party* will apply to the other *Party*'s server authentication certificate.  
2126

2127 The ***ServerSecurityDetailsRef*** element has

- 2128 • A REQUIRED ***securityId*** attribute.  
2129

#### 2130 **8.4.31 Encryption Algorithm**

2131 Zero or more ***EncryptionAlgorithm*** elements may be included under the  
2132 ***TransportClientSecurity*** or ***TransportServerSecurity*** element. Multiple elements are of more  
2133 use in a CPP context, to announce capabilities or preferences; normally, a CPA will contain the  
2134 agreed upon context. When zero or more than one element is present in a CPA, the parties agree  
2135 to allow the automatic negotiation capability of the ***TransportSecurityProtocol*** element to  
2136 determine the actual algorithm used.  
2137

2138 The elements' ordering will reflect the preference for algorithms. A primary reason for including  
2139 this element is to permit use of the ***minimumStrength*** attribute; a large value for this attribute  
2140 can indicate that high encryption strength is desired or has been agreed upon for the  
2141 ***TransportSecurityProtocol***.  
2142

2143 See section 8.4.49 for the full description of this element.  
2144

2145 For SSL and TLS, it is customary to specify cipher suite values as text values for the  
2146 ***EncryptionAlgorithm*** element. These values include, but are not limited to:  
2147

- 2148 • **SSL\_RSA\_FIPS\_WITH\_3DES\_EDE\_CBC\_SHA**,

- 2149 • TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA,
- 2150 • SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA,
- 2151 • SSL\_RSA\_WITH\_RC4\_128\_MD5,
- 2152 • SSL\_RSA\_WITH\_RC4\_128\_SHA,
- 2153 • SSL\_DH\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA,
- 2154 • SSL\_DHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA.

2155 Consult the original specifications for enumerations and discussions of these values.

#### 2156 8.4.32 TransportReceiver element

2157 The **TransportReceiver** element contains properties related to the receiving side of a  
2158 **DeliveryChannel**. Its REQUIRED **TransportProtocol** element specifies the transport protocol  
2159 that will be used for receiving messages. One or more REQUIRED **Endpoint** elements specify  
2160 logical addresses where messages can be received. The **AccessAuthentication** element(s), if  
2161 present, indicates the type(s) of access authentication supported by the server. Zero or one  
2162 **TransportServerSecurity** element defines the *Party*'s provisions for server-side transport layer  
2163 security.

2164 The **TransportReceiver** element has no attributes.

#### 2165 8.4.33 Endpoint element

2166 One or more **Endpoint** elements SHALL be provided for each **TransportReceiver** element. Each  
2167 **Endpoint** specifies a logical address and an indication of what kinds of messages can be received  
2168 at that location.

2169 Each **Endpoint** has the following attributes:

- 2170 • a REQUIRED **uri** attribute,
- 2171 • an IMPLIED **type** attribute.

##### 2172 8.4.33.1 uri attribute

2173 The REQUIRED **uri** attribute specifies a URI identifying the address of a resource. The value of  
2174 the **uri** attribute SHALL conform to the syntax for expressing URIs as defined in [RFC2396].

##### 2175 8.4.33.2 type attribute

2176 The **type** attribute identifies the purpose of this endpoint. The value of **type** is an enumeration;  
2177 permissible values are "login", "request", "response", "error", and "allPurpose". There can be, at  
2178 most, one of each. If the **type** attribute is omitted, its value defaults to "allPurpose". The "login"  
2179 endpoint is used for the address for the initial *Message* between the two *Parties*. The "request"  
2180 and "response" endpoints are used for request and response *Messages*, respectively. To enable  
2181 error *Messages* to be received, each **Transport** element SHALL contain at least one endpoint of  
2182 type "error", "response", or "allPurpose".

2191 The types of **Endpoint** element within a **TransportReceiver** element MUST not be overlapping.  
2192 Thus, it would be erroneous to include both an "allPurpose" **Endpoint** element along with  
2193 another **Endpoint** element of any type.

2194

#### 2195 **8.4.34 TransportServerSecurity element**

2196 The **TransportServerSecurity** element provides information about this *Party*'s transport server  
2197 needed by the other *Party*'s transport client to enable a secure connection to be established  
2198 between the two. It contains a REQUIRED **TransportSecurityProtocol** element, a REQUIRED  
2199 **ServerCertificateRef** element, zero or one **ClientSecurityDetailsRef** element, and zero or more  
2200 **EncryptionAlgorithm** elements. See Section 8.4.31 for a description of the  
2201 **EncryptionAlgorithm** element.

2202

2203     NOTE: See the note in Section 8.4.26 regarding the relevance of the  
2204       **TransportServerSecurity** element when synchronous replies are in use.

2205

#### 2206 **8.4.35 ServerCertificateRef element**

2207 The **ServerCertificateRef** element, if present, identifies the certificate to be used by the server's  
2208 transport security module. The REQUIRED IDREF attribute **certId** identifies the certificate to be  
2209 used by referring to the **Certificate** element (under **PartyInfo**) that has the matching ID attribute  
2210 value. A TLS-enabled HTTP server, for example, uses this certificate to authenticate itself with  
2211 the sender's TLS client.

2212

2213 The **ServerCertificateRef** element MUST be present if the transport security protocol uses  
2214 certificates. It MAY be omitted otherwise (e.g. if authentication is by password).

2215

2216 The **ServerCertificateRef** element has

- A REQUIRED **certId** attribute.

2218

#### 2219 **8.4.36 ClientSecurityDetailsRef element**

2220 The **ClientSecurityDetailsRef** element, if present, identifies the trust anchors and security policy  
2221 that this *Party* will apply to the other *Party*'s client authentication certificate.

2222

2223 The **ClientSecurityDetailsRef** element has

- A REQUIRED **securityId** attribute.

2225

#### 2226 **8.4.37 Transport protocols**

2227 In the following sections, we discuss the specific details of each supported transport protocol.

2228

##### 2229 **8.4.37.1 HTTP**

2230 HTTP is Hypertext Transfer Protocol[HTTP]. For HTTP, the endpoint is a URI that SHALL  
2231 conform to [RFC2396]. Depending on the application, there MAY be one or more endpoints,  
2232 whose use is determined by the application.

2233

2234 Following is an example of an HTTP endpoint:

2235  
2236     <tp:Endpoint tp:uri="http://example.com/servlet/ebxmlhandler"  
2237        tp:type="request"/>

2238  
2239 The "request" and "response" endpoints can be dynamically overridden for a particular request  
2240 or asynchronous response by application-specified URIs in *Business* documents exchanged under  
2241 the *CPA*.

2242  
2243 For a synchronous response, the "response" endpoint is ignored if present. A synchronous  
2244 response is always returned on the existing connection, i.e. to the URI that is identified as the  
2245 source of the connection.

2246  
2247 **8.4.37.2 SMTP**

2248 SMTP is Simple Mail Transfer Protocol[SMTP]. For use with this standard, Multipurpose  
2249 Internet Mail Extensions[MIME] MUST be supported. For SMTP, the communication address is  
2250 the fully qualified mail address of the destination *Party* as defined by [RFC2822]. Following is  
2251 an example of an SMTP endpoint:

2252  
2253     <tp:Endpoint tp:uri="mailto:ebxmlhandler@example.com"  
2254        tp:type="request"/>

2255  
2256 NOTE: The SMTP Mail Transfer Agent (MTA) can encode binary data when the  
2257 receiving MTA does not support binary transfer. In general, SMTP transfer may involve  
2258 coding and recoding of Content-Transfer-Encodings as a message moves along a  
2259 sequence of MTAs. Such changes can in some circumstances invalidate some kinds of  
2260 signatures even though no malicious actions or transmission errors have occurred.

2261  
2262 NOTE: SMTP by itself (without any authentication or encryption) is subject to denial of  
2263 service and masquerading by unknown *Parties*. It is strongly suggested that those *Parties*  
2264 who choose SMTP as their transport layer also choose a suitable means of encryption and  
2265 authentication either in the document-exchange layer or in the transport layer such as  
2266 [S/MIME].

2267  
2268 NOTE: SMTP is an asynchronous protocol that does not guarantee a particular quality of  
2269 service. A transport-layer acknowledgment (i.e. an SMTP acknowledgment) to the  
2270 receipt of a mail *Message* constitutes an assertion on the part of the SMTP server that it  
2271 knows how to deliver the mail *Message* and will attempt to do so at some point in the  
2272 future. However, the *Message* is not hardened and might never be delivered to the  
2273 recipient. Furthermore, the sender will see a transport-layer acknowledgment only from  
2274 the nearest node. If the *Message* passes through intermediate nodes, SMTP does not  
2275 provide an end-to-end acknowledgment. Therefore receipt of an SMTP  
2276 acknowledgement does not guarantee that the *Message* will be delivered to the  
2277 application and failure to receive an SMTP acknowledgment is not evidence that the  
2278 *Message* was not delivered. It is RECOMMENDED that the reliable-messaging protocol  
2279 in the ebXML *Message* Service be used with SMTP.

**2281 8.4.37.3 FTP**

2282 FTP is File Transfer Protocol[RFC959].

2283  
2284 Each *Party* sends a *Message* using FTP PUT. The endpoint specifies the user id and input  
2285 directory path (for PUTs to this *Party*). An example of an FTP endpoint is:

2286  
2287 <tp:Endpoint uri="ftp://userid@server.foo.com"  
2288 tp:type="request"/>

2289  
2290 Since FTP needs to be compatible across all implementations, the FTP for ebXML will use the  
2291 minimum sets of commands and parameters available for FTP as specified in [RFC959], Section  
2292 5.1, and modified in [RFC1123], Section 4.1.2.13. The mode SHALL be stream only and the  
2293 type MUST be ASCII Non-print (AN), Image (I) (binary), or Local 8 (L 8) (binary between 8-bit  
2294 machines and machines with 36 bit words – for an 8-bit machine Local 8 is the same as Image).

2295  
2296 Stream mode closes the data connection upon end of file. The server side FTP MUST set control  
2297 to "PASV" before each transfer command to obtain a unique port pair if there are multiple third  
2298 party sessions.

2299  
2300 NOTE: [RFC 959] states that User-FTP SHOULD send a PORT command to assign a  
2301 non-default data port before each transfer command is issued to allow multiple transfers  
2302 during a single FTP because of the long delay after a TCP connection is closed until its  
2303 socket pair can be reused.

2304  
2305 NOTE: The format of the 227 reply to a PASV command is not well standardized and an  
2306 FTP client might assume that the parentheses indicated in [RFC959] will be present when  
2307 in some cases they are not. If the User-FTP program doesn't scan the reply for the first  
2308 digit of host and port numbers, the result will be that the User-FTP might point at the  
2309 wrong host. In the response, the h1, h2, h3, h4 is the IP address of the server host and the  
2310 p1, p2 is a non-default data transfer port that PASV has assigned.

2311  
2312 NOTE: As a recommendation for firewall transparency, [RFC1579] proposes that the  
2313 client sends a PASV command, allowing the server to do a passive TCP open on some  
2314 random port, and inform the client of the port number. The client can then do an active  
2315 open to establish the connection.

2316  
2317 NOTE: Since STREAM mode closes the data connection upon end of file, the receiving  
2318 FTP might assume abnormal disconnect if a 226 or 250 control code hasn't been received  
2319 from the sending machine.

2320  
2321 NOTE: [RFC1579] also makes the observation that it might be worthwhile to enhance the  
2322 FTP protocol to have the client send a new command APSV (all passive) at startup that  
2323 would allow a server that implements this option to always perform a passive open. A  
2324 new reply code 151 would be issued in response to all file transfer requests not preceded  
2325 by a PORT or PASV command; this *Message* would contain the port number to use for  
2326 that transfer. A PORT command could still be sent to a server that had previously  
2327 received APSV; that would override the default behavior for the next transfer operation,

2328 thus permitting third-party transfers.  
 2329

2330 **8.4.38 DocExchange Element**

2331 The ***DocExchange*** element provides information that the *Parties* MUST agree on regarding  
 2332 exchange of documents between them. This information includes the messaging service  
 2333 properties (e.g. ebXML Message Service[ebMS]).

2334 Following is the structure of the ***DocExchange*** element of the *CPP*. Subsequent sections  
 2335 describe each child element in greater detail.

```
2337
2338   <tp:DocExchange tp:docExchangeId="docExchangeB1">
2339     <tp:ebXMLSenderBinding tp:version="2.0">      <!-- 0 or 1 -->
2340       <tp:ReliableMessaging>                      <!-- 0 or 1 -->
2341       . . .
2342       </tp:ReliableMessaging>
2343       <tp:PersistDuration>                         <!-- 0 or 1 -->
2344       . . .
2345       </tp:PersistDuration>
2346       <tp:SenderNonRepudiation>                  <!-- 0 or 1 -->
2347       . . .
2348       </tp:SenderNonRepudiation>
2349       <tp:SenderDigitalEnvelope>                  <!-- 0 or 1 -->
2350       . . .
2351       </tp:SenderDigitalEnvelope>
2352       <tp:NamespaceSupported>                     <!-- 0 or more -->
2353       . . .
2354       </tp:NamespaceSupported>
2355     </tp:ebXMLSenderBinding>
2356     <tp:ebXMLReceiverBinding tp:version="2.0">    <!-- 0 or 1 -->
2357       <tp:ReliableMessaging>                      <!-- 0 or 1 -->
2358       . . .
2359       </tp:ReliableMessaging>
2360       <tp:PersistDuration>                         <!-- 0 or 1 -->
2361       . . .
2362       </tp:PersistDuration>
2363       <tp:ReceiverNonRepudiation>                 <!-- 0 or 1 -->
2364       . . .
2365       </tp:ReceiverNonRepudiation>
2366       <tp:ReceiverDigitalEnvelope>                <!-- 0 or 1 -->
2367       . . .
2368       </tp:ReceiverDigitalEnvelope>
2369       <tp:NamespaceSupported>                     <!-- 0 or more -->
2370       . . .
2371       </tp:NamespaceSupported>
2372     </tp:ebXMLReceiverBinding>
2373   </tp:DocExchange>
```

2374 The ***DocExchange*** element is comprised of zero or one ***ebXMLSenderBinding*** child element  
 2375 and zero or one ***ebXMLReceiverBinding*** child element. It MUST have at least one child  
 2376 element. *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL verify the  
 2377 presence of a child element.

2378  
 2379 NOTE: The document-exchange section can be extended to messaging services other  
 2380 than the ebXML Message service by adding additional ***xxxSenderBinding*** and  
 2381 ***xxxReceiverBinding*** elements and their child elements that describe the other services,  
 2382 where ***xxx*** is replaced by the name of the additional binding. An example is

2384       ***XMLPSenderBinding/XMLPReceiverBinding***, which might define support for the future  
2385        XML Protocol specification.

2386

#### 2387       **8.4.38.1 docExchangeId attribute**

2388       The ***DocExchange*** element has a single REQUIRED ***docExchangeId*** attribute that is an [XML]  
2389       ID that provides a unique identifier that can be referenced from elsewhere within the *CPP*  
2390       document.

2391

#### 2392       **8.4.39 ebXMLSenderBinding element**

2393       The ***ebXMLSenderBinding*** element describes properties related to sending messages with the  
2394       ebXML Message Service[ebMS]. The ***ebXMLSenderBinding*** element is comprised of the  
2395       following child elements:

- 2396       • zero or one ***ReliableMessaging*** element which specifies the characteristics of reliable  
2397        messaging,
- 2398       • zero or one ***PersistDuration*** element which specifies the duration for which certain  
2399        messages have to be stored persistently for the purpose of duplicate elimination,
- 2400       • zero or one ***SenderNonRepudiation*** element which specifies the sender's  
2401        requirements and certificate for message signing,
- 2402       • zero or one ***SenderDigitalEnvelope*** element which specifies the sender's  
2403        requirements for encryption by the digital-envelope[DIGENV] method,
- 2404       • zero or more ***NamespaceSupported*** elements that identify any namespace extensions  
2405        supported by the messaging service implementation.

2406

2407       The ***ebXMLSenderBinding*** element has one attribute:

- 2408       • a REQUIRED ***version*** attribute.

2409

#### 2410       **8.4.39.1 version attribute**

2411       The REQUIRED ***version*** attribute identifies the version of the ebXML Message Service  
2412       specification being used.

2413

#### 2414       **8.4.40 ReliableMessaging element**

2415       The ***ReliableMessaging*** element specifies the properties of reliable ebXML Message exchange.  
2416       The default that applies if the ***ReliableMessaging*** element is omitted is "BestEffort". The  
2417       following is the element structure:

```
2418       <tp:ReliableMessaging>
2419        <tp:Retries>5</tp:Retries>
2420        <tp:RetryInterval>PT2H</tp:RetryInterval>
2421        <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
2422       </tp:ReliableMessaging>
```

2423

2424       Semantics of reliable messaging are explained in the ebXML Message Service  
2425       specification[ebMS] chapter on Reliable Messaging Combinations.

2426

2427       The ***ReliableMessaging*** element is comprised of the following child elements.

- 2428       • zero or one ***Retries*** element,

- 2430       • zero or one ***RetryInterval*** element,  
2431       • a REQUIRED ***MessageOrderSemantics*** element.

2432

#### 2433     **8.4.40.1 Retries and RetryInterval elements**

2434     The ***Retries*** and ***RetryInterval*** elements specify the permitted number of retries and the interval,  
2435     expressed as an XML Schema[XMLSCHEMA-2] duration, between retries of sending a reliably  
2436     delivered *Message* following a timeout waiting for the *Acknowledgment*. The purpose of the  
2437     ***RetryInterval*** element is to improve the likelihood of success on retry by deferring the retry until  
2438     any temporary conditions that caused the error might be corrected. The *RetryInterval* applies to  
2439     the time between sending of the original message and the first retry, as well as the time between  
2440     all subsequent retries.

2441

2442     The ***Retries*** and ***RetryInterval*** elements MUST either be included together or be omitted  
2443     together. If they are omitted, the values of the corresponding quantities (number of retries and  
2444     retry interval) are a local matter at each *Party*.

2445

#### 2446     **8.4.40.2 MessageOrderSemantics element**

2447     The ***MessageOrderSemantics*** element is an enumeration comprised of the following possible  
2448     values:

- 2449       • "Guaranteed"  
2450       • "NotGuaranteed"

2451

2452     The presence of a ***MessageOrderSemantics*** element in the SOAP Header for ebXML messages  
2453     determines if the ordering of messages sent from the *From Party* needs to be preserved so that  
2454     the *To Party* receives those messages in the order in which they were sent. If the  
2455     ***MessageOrderSemantics*** element is set to "Guaranteed", then the ebXML message MUST  
2456     contain a ***MessageOrder*** element in the SOAP Header. If the ***MessageOrderSemantics*** element  
2457     is set to "NotGuaranteed", then the ebXML message MUST NOT contain a ***MessageOrder***  
2458     element in the SOAP Header. Guaranteed message ordering implies the use of duplicate  
2459     elimination. Therefore, the ***PersistDuration*** element MUST also appear if  
2460     ***MessageOrderSemantics*** is set to "Guaranteed".

2461

#### 2462     **8.4.41 PersistDuration element**

2463     The value of the ***PersistDuration*** element is the minimum length of time, expressed as an XML  
2464     Schema[XMLSCHEMA-2] duration, that data from a *Message* that is sent reliably is kept in  
2465     *Persistent Storage* by an ebXML *Message-Service* implementation that receives that *Message* to  
2466     facilitate the elimination of duplicates. This duration also applies to response messages that are  
2467     kept persistently to allow automatic replies to duplicate messages without their repeated  
2468     processing by the application. For rules that govern the ***PersistDuration*** element, refer to  
2469     Sections 8.4.22.4 and 8.4.40.2.

2470

#### 2471     **8.4.42 SenderNonRepudiation element**

2472     The ***SenderNonRepudiation*** element conveys the message sender's requirements and certificate  
2473     for non-repudiation. Non-repudiation both proves who sent a *Message* and prevents later

2474 repudiation of the contents of the *Message*. Non-repudiation is based on signing the *Message*  
2475 using XML Digital Signature[XMLDSIG]. The element structure is as follows:

```
2476 <tp:SenderNonRepudiation>
2477   <tp:NonRepudiationProtocol>
2478     http://www.w3.org/2000/09/xmldsig#
2479   </tp:NonRepudiationProtocol>
2480   <tp:HashFunction>
2481     http://www.w3.org/2000/09/xmldsig#sha1
2482   </tp:HashFunction>
2483   <tp:SignatureAlgorithm>
2484     http://www.w3.org/2000/09/xmldsig#dsa-sha1
2485   </tp:SignatureAlgorithm>
2486   <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert" />
2487 </tp:SenderNonRepudiation>
```

2490 If the **SenderNonRepudiation** element is omitted, the *Messages* are not digitally signed.

2491

2492 The **SenderNonRepudiation** element is comprised of the following child elements:

- 2493 • a REQUIRED **NonRepudiationProtocol** element,
- 2494 • a REQUIRED **HashFunction** (e.g. SHA1, MD5) element,
- 2495 • a REQUIRED **SignatureAlgorithm** element,
- 2496 • a REQUIRED **SigningCertificateRef** element

2497

#### 2498 8.4.43 NonRepudiationProtocol element

2499 The REQUIRED **NonRepudiationProtocol** element identifies the technology that will be used to  
2500 digitally sign a *Message*. It has a single IMPLIED **version** attribute whose value is a string that  
2501 identifies the version of the specified technology.

2502

#### 2503 8.4.44 HashFunction element

2504 The REQUIRED **HashFunction** element identifies the algorithm that is used to compute the  
2505 digest of the *Message* being signed.

2506

#### 2507 8.4.45 SignatureAlgorithm element

2508 The REQUIRED **SignatureAlgorithm** element identifies the algorithm that is used to compute  
2509 the value of the digital signature. Expected values include: RSA-MD5, RSA-SHA1, DSA-MD5,  
2510 DSA-SHA1, SHA1withRSA, MD5withRSA, and so on.

2511

2512     NOTE: Implementations should be prepared for values in upper and/or lower case and  
2513       with varying usage of hyphens and conjunctions.

2514

2515 The **SignatureAlgorithm** element has three attributes:

- 2516 • an IMPLIED **oid** attribute,
- 2517 • an IMPLIED **w3c** attribute,
- 2518 • an IMPLIED **enumeratedType** attribute.

2519

**2520 8.4.45.1 oid attribute**

2521 The **oid** attribute serves as a way to supply an object identifier for the signature algorithm. The  
2522 formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the  
2523 assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D of  
2524 X.208 (<http://www.itu.int/POD/>). Commonly used values (in the IETF dotted integer format) for  
2525 signature algorithms include:

- 2526 • 1.2.840.113549.1.1.4 - MD5 with RSA encryption,
- 2527 • 1.2.840.113549.1.1.5 - SHA-1 with RSA Encryption.

**2529 8.4.45.2 w3c attribute**

2530 The **w3c** attribute serves as a way to supply an object identifier for the signature algorithm. The  
2531 definitions of these values are found in the [XMLDSIG] or [XMLENC] specifications. Expected  
2532 values for signature algorithms include:

- 2533 • <http://www.w3.org/2000/09/xmldsig#dsa-sha1>,
- 2534 • <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

**2536 8.4.45.3 enumeratedType attribute**

2537 The **enumeratedType** attribute specifies a different way of interpreting the text value of the  
2538 **SignatureAlgorithm** element. This attribute is for identifying future signature algorithm  
2539 identification schemes and formats.

**2541 8.4.46 SigningCertificateRef element**

2542 The REQUIRED **SigningCertificateRef** element identifies the certificate the sender uses for  
2543 signing messages. Its REQUIRED IDREF attribute, **certId** refers to the **Certificate** element  
2544 (under **PartyInfo**) that has the matching ID attribute value.

**2546 8.4.47 SenderDigitalEnvelope element**

2547 The **SenderDigitalEnvelope** element provides the sender's requirements for message encryption  
2548 using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in which the  
2549 *Message* is encrypted by symmetric encryption (shared secret key) and the secret key is sent to  
2550 the *Message* recipient encrypted with the recipient's public key. The element structure is:

```
2551 <tp:SenderDigitalEnvelope>
2552   <tp:DigitalEnvelopeProtocol tp:version="2.0">
2553     S/MIME
2554   </tp:DigitalEnvelopeProtocol>
2555   <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
2556   <tp:EncryptionSecurityDetailsRef
2557     tp:securityId="CompanyA_MessageSecurity" />
2558   </tp:SenderDigitalEnvelope>
```

2560 The **SenderDigitalEnvelope** element contains

- 2561 • a REQUIRED **DigitalEnvelopeProtocol** element,
- 2562 • a REQUIRED **EncryptionAlgorithm** element
- 2563 • zero or one **EncryptionSecurityDetailsRef** element.

2566   **8.4.48 DigitalEnvelopeProtocol element**

2567   The REQUIRED **DigitalEnvelopeProtocol** element identifies the message encryption protocol to  
2568   be used. The REQUIRED **version** attribute identifies the version of the protocol.

2570   **8.4.49 EncryptionAlgorithm element**

2571   The REQUIRED **EncryptionAlgorithm** element identifies the encryption algorithm to be used.  
2572   See also Section 8.4.31.

2573

2574   The **EncryptionAlgorithm** element has four attributes:

- 2575   • an IMPLIED **minimumStrength** attribute,
- 2576   • an IMPLIED **oid** attribute,
- 2577   • an IMPLIED **w3c** attribute,
- 2578   • an IMPLIED **enumeratedType** attribute.

2579

2580   **8.4.49.1 minimumStrength attribute**

2581   The **minimumStrength** attribute describes the effective strength the encryption algorithm MUST  
2582   provide in terms of "effective" or random bits. This value is less than the key length in bits when  
2583   check bits are used in the key. So, for example, the 8 check bits of a 64-bit DES key would not  
2584   be included in the count, and to require a minimum strength the same as that supplied by DES  
2585   would be reported by setting **minimumStrength** to 56.

2586

2587   **8.4.49.2 oid attribute**

2588   The **oid** attribute serves as a way to supply an object identifier for the encryption algorithm. The  
2589   formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the  
2590   assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D of  
2591   X.208 (<http://www.itu.int/POD/>). Commonly used values (in the IETF dotted integer format) for  
2592   encryption algorithms include:

- 2593   • 1.2.840.113549.3.2 (RC2-CBC), 1.2.840.113549.3.4 (RC4 Encryption Algorithm ),
- 2594   • 1.2.840.113549.3.7 (DES-EDE3-CBC ), 1.2.840.113549.3.9 (RC5 CBC Pad),
- 2595   • 1.2.840.113549.3.10 (DES CDMF), 1.2.840.1.3.14.3.2.7 (DES-CBC).

2596

2597   **8.4.49.3 w3c attribute**

2598   The **w3c** attribute serves as a way to supply an object identifier for the encryption algorithm. The  
2599   definitions of these values are in the [XMLENC] specification. Expected values include:

- 2600   • <http://www.w3.org/2001/04/xmlenc#3des-cbc>,
- 2601   • <http://www.w3.org/2001/04/xmlenc#aes128-cbc>,
- 2602   • <http://www.w3.org/2001/04/xmlenc#aes256-cbc>.

2603

2604   **8.4.49.4 enumeratedTypeAttribute**

2605   The **enumeratedType** attribute specifies a way of interpreting the text value of the  
2606   **EncryptionAlgorithm** element. This attribute is for identifying future algorithm identification  
2607   schemes and formats.

2608

2609   **8.4.50 EncryptionSecurityDetailsRef element**

2610   The **EncryptionSecurityDetailsRef** element identifies the trust anchors and security policy that  
2611   this (sending) *Party* will apply to the other (receiving) *Party*'s encryption certificate. Its  
2612   REQUIRED IDREF attribute, **securityId**, refers to the **SecurityDetails** element (under **PartyInfo**)  
2613   that has the matching ID attribute value.

2614

2615   **8.4.51 NamespaceSupported element**

2616   The **NamespaceSupported** element identifies the namespaces supported by the messaging  
2617   service implementation. It has a REQUIRED **location** attribute and an IMPLIED **version**  
2618   attribute. While the **NamespaceSupported** element can be used to list the namespaces that could  
2619   be expected to be used during document exchange, the motivation is primarily for extensions,  
2620   version variants, and other enhancements that might not be expected, or have only recently  
2621   emerged into use.

2622

2623   For example, support for Security Assertion Markup Language[SAML] would be defined as  
2624   follows:

2625

```
<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>
```

2631

2632   In addition, the **NamespaceSupported** element can be used to identify the namespaces associated  
2633   with the message body parts (see Section 8.5), and especially when these namespaces are not  
2634   implicitly indicated through parts of the **ProcessSpecification** or when they indicate extensions  
2635   of namespaces for payload body parts.

2636

2637   **8.4.52 ebXMLReceiverBinding element**

2638   The **ebXMLReceiverBinding** element describes properties related to receiving messages with the  
2639   ebXML Message Service[ebMS]. The **ebXMLReceiverBinding** element is comprised of the  
2640   following child elements:

- 2641     • zero or one **ReliableMessaging** element (see Section 8.4.40),
- 2642     • zero or one **ReceiverNonRepudiation** element which specifies the receiver's  
2643        requirements for message signing,
- 2644     • zero or one **ReceiverDigitalEnvelope** element which specifies the receiver's  
2645        requirements and certificate for encryption by the digital-envelope[DIGENV]  
2646        method,
- 2647     • zero or more **NamespaceSupported** elements (see Section 8.4.51).

2648

2649   The **ebXMLReceiverBinding** element has one attribute:

- 2650     • a REQUIRED **version** attribute (see Section 8.4.39.1)

2651

#### 2652    8.4.53 ReceiverNonRepudiation element

2653    The **ReceiverNonRepudiation element** conveys the message receiver's requirements for non-  
2654    repudiation. Non-repudiation both proves who sent a *Message* and prevents later repudiation of  
2655    the contents of the *Message*. Non-repudiation is based on signing the *Message* using XML  
2656    Digital Signature[XMLDSIG]. The element structure is as follows:

```
2657       <tp:ReceiverNonRepudiation>
2658           <tp:NonRepudiationProtocol>
2659             http://www.w3.org/2000/09/xmldsig#
2660           </tp:NonRepudiationProtocol>
2661           <tp:HashFunction>
2662             http://www.w3.org/2000/09/xmldsig#sha1
2663           </tp:HashFunction>
2664           <tp:SignatureAlgorithm>
2665             http://www.w3.org/2000/09/xmldsig#dsa-sha1
2666           </tp:SignatureAlgorithm>
2667           <tp:SigningSecurityDetailsRef tp:certId="CompanyA_MessageSecurity" />
2668       </tp:ReceiverNonRepudiation>
```

2669    If the **ReceiverNonRepudiation** element is omitted, the *Messages* are not digitally signed.

- 2670    • a REQUIRED **NonRepudiationProtocol** element (see Section 8.4.43),
- 2671    • a REQUIRED **HashFunction** (e.g. SHA1, MD5) element (see Section 8.4.44),
- 2672    • a REQUIRED **SignatureAlgorithm** element (see Section 8.4.45),
- 2673    • zero or one **SigningSecurityDetailsRef** element

#### 2674    8.4.54 SigningSecurityDetailsRef element

2675    The **SigningSecurityDetailsRef** element identifies the trust anchors and security policy that this  
2676    (receiving) *Party* will apply to the other (sending) *Party*'s signing certificate. Its REQUIRED  
2677    IDREF attribute, **securityId**, refers to the **SecurityDetails** element (under **PartyInfo**) that has the  
2678    matching ID attribute value.

#### 2679    8.4.55 ReceiverDigitalEnvelope element

2680    The **ReceiverDigitalEnvelope** element provides the receiver's requirements for message  
2681    encryption using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in  
2682    which the *Message* is encrypted by symmetric encryption (shared secret key) and the secret key  
2683    is sent to the *Message* recipient encrypted with the recipient's public key. The element structure  
2684    is:

```
2685       <tp:ReceiverDigitalEnvelope>
2686           <tp:DigitalEnvelopeProtocol tp:version="2.0">
2687             S/MIME
2688           </tp:DigitalEnvelopeProtocol>
2689           <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
2690           <tp:EncryptionCertificateRef
2691             tp:certId="CompanyA_EncryptionCert" />
2692       </tp:ReceiverDigitalEnvelope>
```

2693    The **ReceiverDigitalEnvelope** element contains

- 2702     • a REQUIRED **DigitalEnvelopeProtocol** element (see Section 8.4.48),  
2703     • a REQUIRED **EncryptionAlgorithm** element (see Section 8.4.49),  
2704     • a REQUIRED **EncryptionCertificateRef** element.  
2705

#### 2706     **8.4.56 EncryptionCertificateRef element**

2707     The REQUIRED **EncryptionCertificateRef** element identifies the certificate the sender uses for  
2708     encrypting messages. Its REQUIRED IDREF attribute, **certId** refers to the **Certificate** element  
2709     (under **PartyInfo**) that has the matching ID attribute value.

#### 2711     **8.4.57 OverrideMshActionBinding element**

2712     The **OverrideMshActionBinding** element can occur zero or more times. It has two REQUIRED  
2713     attributes. The **action** attribute identifies the *Message Service Handler* level action whose  
2714     delivery is not to use the default **DeliveryChannel** for *Message Service Handler* actions. The  
2715     channelId attribute specifies the **DeliveryChannel** to be used instead.  
2716

### 2717     **8.5 SimplePart element**

2718     The **SimplePart** element provides a repeatable list of the constituent parts, primarily identified by  
2719     the MIME content-type value. The **SimplePart** element has two REQUIRED attributes: **id** and  
2720     **mimetype**. The **id** attribute, of type ID, provides the value that will be used later to reference this  
2721     *Message* part when specifying how the parts are packaged into composites, if composite  
2722     packaging is present. The **mimetype** attribute can provide actual values of content-type for the  
2723     simple *Message* part being specified. The attribute's values may also make use of an asterisk  
2724     wildcard, “\*”, to indicate either an arbitrary top-level type, an arbitrary sub-type, or a completely  
2725     arbitrary type, “\*/\*”. SimpleParts with wildcards in types can be used in indicating more open  
2726     packaging processing capabilities.  
2727

2728     **SimplePart** also has an IMPLIED **xlink:role** attribute which identifies some resource that  
2729     describes the mime part or its purpose. If present, then it SHALL have a value that is a valid URI  
2730     in accordance with the [XLINK] specification. The following are examples of **SimplePart**  
2731     elements:

2732  
2733       <tp:SimplePart tp:id="I001" tp:mimetype="text/xml"/>  
2734       <tp:SimplePart tp:id="I002" tp:mimetype="application/xml"/>  
2735       <tp:SimplePart tp:id="I002" tp:mimetype="\*/xml"/>  
2736

2737     The **SimplePart** element can have zero or more **NamespaceSupported** elements. Each of these  
2738     identifies any namespace supported for the XML that is packaged in the parent simple body part.  
2739

2740     The context of **Packaging** can very easily render it pointless to list all the namespaces used in a  
2741     **SimplePart**. For example, when defining the **SimplePart** for a SOAP envelope, as part of an  
2742     ebXML Message, it is not necessary to list all the namespaces. If, however, any unusual  
2743     extensions, new versions, or unusual security extensions are present, it is useful to announce  
2744     these departures explicitly in the packaging. It is not, however, incorrect to list all namespaces  
2745     used in a **SimplePart**, even where these namespaces have been mandated by a given messaging

2746 protocol. By convention, when a full listing of namespaces is supplied within a **SimplePart**  
2747 element, the first **NamespaceSupported** element identifies the schema for the **SimplePart** while  
2748 subsequent **NamespaceSupported** elements represent namespaces that are imported by that  
2749 schema. Any additional **NamespaceSupported** elements indicate extensions.  
2750

2751 NOTE: The explicit identification of imported namespaces is discretionary. Thus, the  
2752 CPP and CPA examples in Appendix A and Appendix B explicitly identify the ebXML  
2753 Messaging Service namespace but omit the SOAP envelope and XML Digital Signature  
2754 namespaces that are imported into the schema for the ebXML Messaging Service  
2755 namespace.

2756 The same **SimplePart** element can be referenced from (i.e., reused in) multiple **Packaging**  
2757 elements.  
2759

## 2760 8.6 Packaging element

2761 The subtree of the **Packaging** element provides specific information about how the *Message*  
2762 *Header* and payload constituent(s) are packaged for transmittal over the transport, including the  
2763 crucial information about what document-level security packaging is used and the way in which  
2764 security features have been applied. Typically the subtree under the **Packaging** element indicates  
2765 the specific way in which constituent parts of the *Message* are organized. MIME processing  
2766 capabilities are typically the capabilities or agreements described in this subtree. The **Packaging**  
2767 element provides information about MIME content types, XML namespaces, security  
2768 parameters, and MIME structure of the data that is exchanged between *Parties*.  
2769

2770 The following is an example of a **Packaging** element which references the example **SimplePart**  
2771 elements given in Section 8.5:

```
2772 <!-- Simple ebXML S/MIME Packaging for application-based payload  
2773     encryption -->  
2774 <tp:Packaging>  
2775     <tp:ProcessingCapabilities tp:generate="true" tp:parse="true"/>  
2776     <tp:CompositeList>  
2777         <tp:Encapsulation  
2778             <!-- I002 is the payload being encrypted -->  
2779             tp:id="I003"  
2780             tp:mimetype="application/pkcs7-mime"  
2781             tp:mimeparameters="smime-type="text/enveloped-data""/>  
2782             <Constituent tp:idref="I002"/>  
2783         </tp:Encapsulation>  
2784         <tp:Composite tp:id="I004"  
2785             <!-- I001 is the SOAP envelope. The ebXML message is made  
2786                 up of the SOAP envelope and the encrypted payload. -->  
2787             tp:mimetype="multipart/related"  
2788             tp:mimeparameters="type="text/xml";  
2789             version="1.0">  
2790             <tp:Constituent tp:idref="I001"/>  
2791             <tp:Constituent tp:idref="I003"/>  
2792         </tp:Composite>  
2793     </tp:CompositeList>  
2794 </tp:Packaging>
```

2797 The **Packaging** element has one attribute; the REQUIRED *id* attribute, with type ID. It is  
2798 referred to in the **ThisPartyActionBinding** element, by using the IDREF attribute, *packageId*.  
2799

2800 The child elements of the **Packaging** element are **ProcessingCapabilities** and **CompositeList**.  
2801 This set of elements can appear one or more times as a child of each **Packaging** element.  
2802

### 2803 **8.6.1 ProcessingCapabilities element**

2804 The **ProcessingCapabilities** element has two REQUIRED attributes with Boolean values of  
2805 either "true" or "false". The attributes are *parse* and *generate*. Normally, these attributes will  
2806 both have values of "true" to indicate that the packaging constructs specified in the other child  
2807 elements can be both produced as well as processed at the software *Message* service layer.  
2808 At least one of the *generate* or *parse* attributes MUST be true.  
2809

### 2810 **8.6.2 CompositeList element**

2811 The final child element of **Packaging** is **CompositeList**, which is a container for the specific way  
2812 in which the simple parts are combined into groups (MIME multipart) or encapsulated within  
2813 security-related MIME content-types. The **CompositeList** element SHALL be omitted from  
2814 **Packaging** when no security encapsulations or composite multipart are used. When the  
2815 **CompositeList** element is present, the content model for the **CompositeList** element is a  
2816 repeatable sequence of choices of **Composite** or **Encapsulation** elements. The **Composite** and  
2817 **Encapsulation** elements can appear intermixed as desired. The sequence in which the choices  
2818 are presented is important because, given the recursive character of MIME packaging,  
2819 composites or encapsulations can include previously mentioned composites (or rarely,  
2820 encapsulations) in addition to the *Message* parts characterized within the **SimplePart** subtree.  
2821 Therefore, the "top-level" packaging will be described last in the sequence.  
2822

2823 The **Composite** element has the following attributes:

- 2824 • a REQUIRED *mimetype* attribute,
- 2825 • a REQUIRED *id* attribute,
- 2826 • an IMPLIED *mimeparameters* attribute.

2827 The *mimetype* attribute provides the value of the MIME content-type for this *Message* part, and  
2828 this will be some MIME composite type, such as "multipart/related" or "multipart/signed". The  
2829 *id* attribute, type ID, provides a way to refer to this composite if it needs to be mentioned as a  
2830 constituent of some later element in the sequence. The *mimeparameters* attribute provides the  
2831 values of any significant MIME parameter (such as "type=application/xml") that is needed to  
2832 understand the processing demands of the content-type.  
2833

2834 The **Composite** element has one child element, **Constituent**.

2835 The **Constituent** element has one REQUIRED attribute, *idref* of type IDREF, an IMPLIED  
2836 boolean attribute *excludeFromSignature*, and two IMPLIED nonNegativeInteger attributes,  
2837 *minOccurs* and *maxOccurs*.  
2838

2841 The *idref* attribute has as its value the value of the *id* attribute of a previous *Composite*,  
2842 *Encapsulation*, or *SimplePart* element. The purpose of this sequence of *Constituents* is to  
2843 indicate both the contents and the order of what is packaged within the current *Composite* or  
2844 *Encapsulation*.

2845  
2846 The *excludeFromSignature* attribute indicates that this Constituent is not to be included as part  
2847 of the ebXML message [XMLDSIG] signature. In other words, the signature generated by the  
2848 *Message Service Handler* should not include a *ds:Reference* element to provide a digest for this  
2849 *Constituent* of the *Message*. This attribute is applicable only if the *Constituent* is part of the top-  
2850 level *Composite* that corresponds to the entire ebXML *Message*.

2851  
2852 The *minOccurs* and *maxOccurs* attributes serve to specify the value or range of values that the  
2853 referred to item may occur within *Composite*. When unused, it is understood that the item is used  
2854 exactly once.

2855  
2856 The *Encapsulation* element is typically employed to indicate the use of MIME security  
2857 mechanisms, such as [S/MIME] or Open-PGP[RFC2015]. A security body part can encapsulate a  
2858 MIME part that has been previously characterized. For convenience, all such security structures  
2859 are under the *Encapsulation* element, even when technically speaking the data is not "inside" the  
2860 body part. (In other words, the so-called clear-signed or detached signature structures possible  
2861 with MIME multipart/signed are for simplicity found under the *Encapsulation* element.)

2862  
2863 Another possible use of the *Encapsulation* element is to represent the application of a  
2864 compression algorithm such as gzip [ZLIB] to some part of the payload, prior to its being  
2865 encrypted and or signed.

2866  
2867 The *Encapsulation* element has the following attributes:

- 2868 • a REQUIRED *mimetype* attribute,
- 2869 • a REQUIRED *id* attribute,
- 2870 • an IMPLIED *mimeparameters* attribute.

2871  
2872 The *mimetype* attribute provides the value of the MIME content-type for this *Message* part, such  
2873 as "application/pkcs7-mime". The *id* attribute, type ID, provides a way to refer to this  
2874 encapsulation if it needs to be mentioned as a constituent of some later element in the sequence.  
2875 The *mimeparameters* attribute provides the values of any significant MIME parameter(s)  
2876 needed to understand the processing demands of the content-type.

2877  
2878 Both the *Encapsulation* element and the *Composite* element have child elements consisting of a  
2879 *Constituent* element or of a repeatable sequence of *Constituent* elements, respectively.

2880  
2881 The *Constituent* element also has zero or one *SignatureTransform* child element and zero or  
2882 one *EncryptionTransform* child element. The *SignatureTransform* element is intended for use  
2883 with XML Digital Signature [XMLDSIG]. When present, it identifies the transforms that must  
2884 be applied to the source data before a digest is computed. The *EncryptionTransform* element is  
2885 intended for use with XML Encryption [XMLENC]. When present, it identifies the transforms  
2886 that must be applied to a *CipherReference* before decryption can be performed. The

2887     ***SignatureTransforms*** element and the ***EncryptionTransforms*** element each contains one or  
2888     more ***ds:Transform*** [XMLDSIG] elements.  
2889

## 2890     **8.7 Signature element**

2891     The ***Signature*** element (cardinality zero or one) enables the CPA to be digitally signed using  
2892     technology that conforms with the XML Digital Signature specification[XMLDSIG]. The  
2893     ***Signature*** element is the root of a subtree of elements used for signing the *CPP*. The syntax is:  
2894

```
2895       <tp:Signature>...</tp:Signature>
```

2896  
2897     The ***Signature*** element contains one or more ***ds:Signature*** elements. The content of the  
2898     ***ds:Signature*** element and any sub-elements are defined by the XML Digital Signature  
2899     specification. See Section 9.9 for a detailed discussion.

2900  
2901     NOTE: It is necessary to wrap the ***ds:Signature*** elements with a ***Signature*** element in the  
2902     target namespace to allow for the possibility of having wildcard elements (with  
2903     namespace="#other") within the *CollaborationProtocolProfile* and  
2904     *CollaborationProtocolAgreement* elements. The content model would be ambiguous without  
2905     the wrapping.  
2906

2907     The following additional constraints on ***ds:Signature*** are imposed:  
2908

- 2909     • A *CPP* MUST be considered invalid if any ***ds:Signature*** element fails core validation as  
2910       defined by the XML Digital Signature specification[XMLDSIG].  
2911
- 2912     • Whenever a *CPP* is signed, each ***ds:Reference*** element within a ***ProcessSpecification***  
2913       element MUST pass reference validation and each ***ds:Signature*** element MUST pass  
2914       core validation.  
2915

2916     NOTE: In case a *CPP* is unsigned, software might nonetheless validate the ***ds:Reference***  
2917     elements within ***ProcessSpecification*** elements and report any exceptions.  
2918

2919     NOTE: Software for creation of *CPPs* and *CPAs* MAY recognize ***ds:Signature*** and  
2920     automatically insert the element structure necessary to define signing of the *CPP* and *CPA*.  
2921     Signature generation is outlined in Section 9.9.1.1; details of the cryptographic process are  
2922     outside the scope of this specification.  
2923

2924     NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which validity  
2925     tests MAY be made.  
2926

## 2927     **8.8 Comment element**

2928     The ***CollaborationProtocolProfile*** element contains zero or more ***Comment*** elements. The  
2929     ***Comment*** element is a textual note that can be added to serve any purpose the author desires.  
2930     The language of the ***Comment*** is identified by a REQUIRED ***xml:lang*** attribute. The ***xml:lang***  
2931     attribute MUST comply with the rules for identifying languages specified in [XML]. If multiple

2932     **Comment** elements are present, each can have a different *xml:lang* attribute value. An example  
2933     of a **Comment** element follows:

2934       <tp:Comment xml:lang="en-US">This is a CPA between A and B</tp:Comment>

2935     When a *CPA* is composed from two *CPPs*, all **Comment** elements from both *CPPs* SHALL be  
2936     included in the *CPA* unless the two *Parties* agree otherwise.

## 2939 **9 CPA Definition**

2940 A *Collaboration-Protocol Agreement (CPA)* defines the capabilities that two *Parties* need to  
2941 agree upon to enable them to engage in electronic *Business* for the purposes of the particular  
2942 *CPA*. This section defines and discusses the details of the *CPA*. The discussion is illustrated with  
2943 some XML fragments.

2944 Most of the XML elements in this section are described in detail in Section 8, "CPP Definition".  
2945 In general, this section does not repeat that information. The discussions in this section are  
2946 limited to those elements that are not in the *CPP* or for which additional discussion is needed in  
2947 the *CPA* context. See also Appendix D for the XML Schema, and Appendix B for an example of  
2948 a *CPA* document.

2950

### 2951 **9.1 CPA Structure**

2952 Following is the overall structure of the *CPA*:

2953

```
2954 <CollaborationProtocolAgreement
2955   xmlns:tp="http://www.oasis-open.org/committees/ebxml-
2956   cппa/schema/cpp-cpa-2_0.xsd"
2957   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2958   xmlns:xlink="http://www.w3.org/1999/xlink"
2959   tp:cpaid="YoursAndMyCPA"
2960   tp:version="2.0a">
2961   <tp>Status tp:value="proposed"/>
2962   <tp:Start>1988-04-07T18:39:09</Start>
2963   <tp:End>1990-04-07T18:40:00</End>
2964   <!-- ConversationConstraints MAY appear 0 or 1 time -->
2965   <tp:ConversationConstraints
2966     tp:invocationLimit="100"
2967     tp:concurrentConversations="4" />
2968   <tp:PartyInfo>
2969   ...
2970   </tp:PartyInfo>
2971   <tp:PartyInfo>
2972   ...
2973   </tp:PartyInfo>
2974   <tp:SimplePart tp:id="..."><!-- one or more -->
2975   ...
2976   </tp:SimplePart>
2977   <tp:Packaging tp:id="..."><!-- one or more -->
2978   ...
2979   </tp:Packaging>
2980   <tp:Signature><!-- zero or one time -->
2981   ...
2982   </tp:Signature>
2983   <tp:Comment xml:lang="en-GB">any text</Comment><!-- zero or more --
2984   >
2985   </tp:CollaborationProtocolAgreement>
```

## 2987 9.2 CollaborationProtocolAgreement element

2988 The **CollaborationProtocolAgreement** element is the root element of a *CPA*. It has a  
2989 REQUIRED *cpaid* attribute that supplies a unique identifier for the document. The value of the  
2990 *cpaid* attribute SHALL be assigned by one *Party* and used by both. It is RECOMMENDED that  
2991 the value of the *cpaid* attribute be a URI. The value of the *cpaid* attribute SHALL be used as the  
2992 value of the **CPAId** element in the ebXML *Message Header*[ebMS] or of a similar element in a  
2993 *Message Header* of an alternative messaging service.

2994  
2995       NOTE: Each *Party* might associate a local identifier with the *cpaid* attribute.  
2996

2997 In addition, the **CollaborationProtocolAgreement** element has a REQUIRED *version* attribute.  
2998 This attribute indicates the version of the schema to which the *CPA* conforms. The value of the  
2999 *version* attribute SHOULD be a string such as "2\_0a", "2\_0b", etc.

3000       NOTE: The method of assigning unique *cpaid* values is left to the implementation.  
3001

3002 The **CollaborationProtocolAgreement** element has REQUIRED [XML] Namespace[XMLNS]  
3003 declarations that are defined in Section 8, "CPP Definition".  
3004

3005 The **CollaborationProtocolAgreement** element is comprised of the following child elements,  
3006 most of which are described in greater detail in subsequent sections:

- 3007     • a REQUIRED **Status** element that identifies the state of the process that creates the  
3008       *CPA*,
- 3009     • a REQUIRED **Start** element that records the date and time that the *CPA* goes into  
3010       effect,
- 3011     • a REQUIRED **End** element that records the date and time after which the *CPA*  
3012       MUST be renegotiated by the *Parties*,
- 3013     • zero or one **ConversationConstraints** element that documents certain agreements  
3014       about conversation processing,
- 3015     • two REQUIRED **PartyInfo** elements, one for each *Party* to the *CPA*,
- 3016     • one or more **SimplePart** elements,
- 3017     • one or more **Packaging** elements,
- 3018     • zero or one **Signature** element that provides for signing of the *CPA* using the XML  
3019       Digital Signature[XMLDSIG] standard,
- 3020     • zero or more **Comment** elements.

## 3022 9.3 Status Element

3023 The **Status** element records the state of the composition/negotiation process that creates the *CPA*.  
3024 An example of the **Status** element follows:

3025  
3026       <tp:Status tp:value="proposed" />  
3027

3028 The Status element has a REQUIRED *value* attribute that records the current state of  
3029 composition of the *CPA*. This attribute is an enumeration comprised of the following possible  
3030 values:

- 3031     • "proposed", meaning that the *CPA* is still being negotiated by the *Parties*,

- 3032     • "agreed", meaning that the contents of the *CPA* have been agreed to by both *Parties*,  
3033     • "signed", meaning that the *CPA* has been "signed" by the *Parties*. This "signing"  
3034        takes the form of a digital signature that is described in Section 9.7 below.

3035  
3036        NOTE: The **Status** element MAY be used by a *CPA* composition and negotiation tool to  
3037        assist it in the process of building a *CPA*.

## 3039        **9.4 CPA Lifetime**

3040        The lifetime of the *CPA* is given by the **Start** and **End** elements. The syntax is:

3041  
3042           <tp:Start>1988-04-07T18:39:09Z</tp:Start>  
3043           <tp:End>1990-04-07T18:40:00Z</tp:End>

### 3045        **9.4.1 Start element**

3046        The **Start** element specifies the starting date and time of the *CPA*. The **Start** element SHALL be  
3047        a string value that conforms to the content model of a canonical dateTime type as defined in the  
3048        XML Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm  
3049        UTC (Coordinated Universal Time) on May 31, 1999, a **Start** element would have the following  
3050        value:

3051  
3052           1999-05-31T13:20:00Z

3053  
3054        The **Start** element SHALL be represented as Coordinated Universal Time (UTC).

### 3056        **9.4.2 End element**

3057        The **End** element specifies the ending date and time of the *CPA*. The **End** element SHALL be a  
3058        string value that conforms to the content model of a canonical dateTime type as defined in the  
3059        XML Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm  
3060        UTC (Coordinated Universal Time) on May 31, 1999, an **End** element would have the following  
3061        value:

3062  
3063           1999-05-31T13:20:00Z

3064  
3065        The **End** element SHALL be represented as Coordinated Universal Time (UTC).

3066  
3067        When the end of the *CPA*'s lifetime is reached, any *Business Transactions* that are still in  
3068        progress SHALL be allowed to complete and no new *Business Transactions* SHALL be started.  
3069        When all in-progress *Business Transactions* on each conversation are completed, the  
3070        *Conversation* SHALL be terminated whether or not it was completed.

3071  
3072        When a *CPA* is signed, software for signing the agreements SHALL warn if any signing  
3073        certificate's validity expires prior to the proposed time for ending the *CPA*. The opportunity to  
3074        renegotiate a *CPA* **End** value or to in some other way align certificate validity periods with *CPA*  
3075        validity periods SHALL be made available. (Other ways to align these validity periods would

3076 include reissuing the signing certificates for a longer period or obtaining new certificates for this  
3077 purpose.)

3078  
3079 Signing software SHOULD also attempt to align the validity periods of certificates referred to  
3080 within the CPA that perform security functions so as to not expire before the CPA expires. This  
3081 alignment can occur in several ways including making use of ***ds:KeyInfo***'s content model  
3082 ***ds:RetrievalMethod*** so that a new certificate can be installed and still be retrieved in accordance  
3083 with the information in ***ds:RetrievalMethod***. If no alignment can be attained, signing software  
3084 MUST warn the user of the situation that the *CPA* validity exceeds the validity of some of the  
3085 certificates referred to within the *CPA*.

3086  
3087 NOTE: If a *Business* application defines a conversation as consisting of multiple *Business*  
3088 *Transactions*, such a conversation MAY be terminated with no error indication when the  
3089 end of the lifetime is reached. The run-time system could provide an error indication to  
3090 the application.

3091  
3092 NOTE: It might not be feasible to wait for outstanding conversations to terminate before  
3093 ending the *CPA* since there is no limit on how long a conversation can last.

3094  
3095 NOTE: The run-time system SHOULD return an error indication to both *Parties* when a  
3096 new *Business Transaction* is started under this *CPA* after the date and time specified in  
3097 the ***End*** element.

3098

## 3099 **9.5 ConversationConstraints Element**

3100 The ***ConversationConstraints*** element places limits on the number of conversations under the  
3101 *CPA*. An example of this element follows:

3102  
3103 <tp:ConversationConstraints tp:invocationLimit="100"  
3104       tp:concurrentConversations="4" />  
3105

3106 The ***ConversationConstraints*** element has the following attributes:

- 3107     • an IMPLIED ***invocationLimit*** attribute,  
3108     • an IMPLIED ***concurrentConversations*** attribute.

3109

### 3110 **9.5.1 invocationLimit attribute**

3111 The ***invocationLimit*** attribute defines the maximum number of conversations that can be  
3112 processed under the *CPA*. When this number has been reached, the *CPA* is terminated and  
3113 MUST be renegotiated. If no value is specified, there is no upper limit on the number of  
3114 conversations and the lifetime of the *CPA* is controlled solely by the ***End*** element.

3115  
3116 NOTE: The ***invocationLimit*** attribute sets a limit on the number of units of *Business* that  
3117 can be performed under the *CPA*. It is a *Business* parameter, not a performance  
3118 parameter.  
3119

3120    **9.5.2 concurrentConversations attribute**

3121    The **concurrentConversations** attribute defines the maximum number of conversations that can  
3122    be in process under this *CPA* at the same time. If no value is specified, processing of concurrent  
3123    conversations is strictly a local matter.

3124    NOTE: The **concurrentConversations** attribute provides a parameter for the *Parties* to use  
3125    when it is necessary to limit the number of conversations that can be concurrently processed  
3126    under a particular *CPA*. For example, the back-end process might only support a limited  
3127    number of concurrent conversations. If a request for a new conversation is received when  
3128    the maximum number of conversations allowed under this *CPA* is already in process, an  
3129    implementation MAY reject the new conversation or MAY enqueue the request until an  
3130    existing conversation ends. If no value is given for **concurrentConversations**, how to handle  
3131    a request for a new conversation for which there is no capacity is a local implementation  
3132    matter.  
3133

3134    **9.6 PartyInfo Element**

3135    The general characteristics of the **PartyInfo** element are discussed in Section 8.4.

3136    The *CPA* SHALL have one **PartyInfo** element for each *Party* to the *CPA*. The **PartyInfo**  
3137    element specifies the *Parties*' agreed terms for engaging in the *Business Collaborations* defined  
3138    by the *Process-Specification* documents referenced by the *CPA*. If a *CPP* has more than one  
3139    **PartyInfo** element, the appropriate **PartyInfo** element SHALL be selected from each *CPP* when  
3140    composing a *CPA*.

3141    In the *CPA*, there SHALL be one or more **PartyId** elements under each **PartyInfo** element. The  
3142    values of these elements are the same as the values of the **PartyId** elements in the ebXML  
3143    *Message Service specification*[ebMS] or similar messaging service specification. These **PartyId**  
3144    elements SHALL be used within a **To** or **From** *Header* element of an ebXML *Message*.  
3145

3146    **9.6.1 ProcessSpecification element**

3147    The **ProcessSpecification** element identifies the *Business Collaboration* that the two *Parties*  
3148    have agreed to perform. There can be one or more **ProcessSpecification** elements in a *CPA*.  
3149    Each SHALL be a child element of a separate **CollaborationRole** element. See the discussion in  
3150    Section 8.4.3.  
3151

3152    **9.7 SimplePart element**

3153    The **CollaborationProtocolAgreement** element SHALL contain one or more **SimplePart**  
3154    elements. See Section 8.5 for details of the syntax of the **SimplePart** element.  
3155

3156    **9.8 Packaging element**

3157    The **CollaborationProtocolAgreement** element SHALL contain one or more **Packaging**  
3158    elements. See Section 8.6 for details of the syntax of the **Packaging** element.  
3159

3162

## 3163 9.9 Signature element

3164 A *CPA* document can be digitally signed by one or more of the *Parties* as a means of ensuring its  
3165 integrity as well as a means of expressing the agreement just as a corporate officer's signature  
3166 would do for a paper document. If signatures are being used to digitally sign an ebXML *CPA* or  
3167 *CPP* document, then [XMLDSIG] SHALL be used to digitally sign the document.

3168

3169 The ***Signature*** element, if present, is made up of one or more ***ds:Signature*** elements. In a *CPA*  
3170 involving two *Parties*, there can be up to three ***ds:Signature*** elements within the ***Signature***  
3171 element. The *CPA* is initially signed by one of the two *Parties*. The other *Party* could then sign  
3172 over the first *Party*'s signature. The resulting *CPA* MAY then be signed by a notary.

3173

3174 The ***ds:Signature*** element is the root of a subtree of elements used for signing the *CPP*.

3175

3176 The content of this element and any sub-elements are defined by the XML Digital Signature  
3177 specification[XMLDSIG]. The following additional constraints on ***ds:Signature*** are imposed:

3178

- A *CPA* MUST be considered invalid if any ***ds:Signature*** fails core validation as defined  
3179 by the XML Digital Signature specification.
- Whenever a *CPA* is signed, each ***ds:Reference*** within a ***ProcessSpecification*** MUST  
3180 pass reference validation and each ***ds:Signature*** MUST pass core validation.

3181

3182 NOTE: In case a *CPA* is unsigned, software MAY nonetheless validate the ***ds:Reference***  
3183 elements within ***ProcessSpecification*** elements and report any exceptions.

3184

3185 Software for creation of *CPPs* and *CPAs* SHALL recognize ***ds:Signature*** and automatically  
3186 insert the element structure necessary to define signing of the *CPP* and *CPA*. Signature creation  
3187 itself is a cryptographic process that is outside the scope of this specification.

3188

3189 NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which a *CPA*  
3190 MAY be validated.

3191

### 3192 9.9.1 Persistent Digital Signature

3193

3194 If [XMLDSIG] is used to sign an ebXML *CPP* or *CPA*, the process defined in this section of the  
specification SHALL be used.

3195

#### 3196 9.9.1.1 Signature Generation

3197 Following are the steps to create a digital signature:

3198

1. Create a ***SignedInfo*** element, a child element of ***ds:Signature***. ***SignedInfo*** SHALL have  
3199 child elements ***SignatureMethod***, ***CanonicalizationMethod***, and ***Reference*** as prescribed by  
3200 [XMLDSIG].
2. Canonicalize and then calculate the ***SignatureValue*** over ***SignedInfo*** based on algorithms

- 3206        specified in **SignedInfo** as specified in [XMLDSIG].  
3207     3. Construct the **Signature** element that includes the **SignedInfo**, **KeyInfo**  
3208        (RECOMMENDED), and **SignatureValue** elements as specified in [XMLDSIG].  
3209     4. Include the namespace qualified **Signature** element in the document just signed, following  
3210        the last **PartyInfo** element.

3211     **9.9.1.2 ds:SignedInfo element**

3212     The **ds:SignedInfo** element SHALL be comprised of zero or one **ds:CanonicalizationMethod**  
3213        element, the **ds:SignatureMethod** element, and one or more **ds:Reference** elements.

3214     **9.9.1.3 ds:CanonicalizationMethod element**

3215     The **ds:CanonicalizationMethod** element as defined in [XMLDSIG], can occur zero or one  
3216        time, meaning that the element need not appear in an instance of a **ds:SignedInfo** element. The  
3217        default canonicalization method that is applied to the data to be signed is [XMLC14N] in the  
3218        absence of a **ds:CanonicalizationMethod** element that specifies otherwise. This default SHALL  
3219        also serve as the default canonicalization method for the ebXML *CPP* and *CPA* documents.

3220     **9.9.1.4 ds:SignatureMethod element**

3221     The **ds:SignatureMethod** element SHALL be present and SHALL have an **Algorithm** attribute.  
3222     The RECOMMENDED value for the **Algorithm** attribute is:

3223        "http://www.w3.org/2000/09/xmldsig#sha1"

3224     This RECOMMENDED value SHALL be supported by all compliant ebXML *CPP* or *CPA*  
3225        software implementations.

3226     **9.9.1.5 ds:Reference element**

3227     The **ds:Reference** element for the *CPP* or *CPA* document SHALL have a REQUIRED URI  
3228        attribute value of "" to provide for the signature to be applied to the document that contains the  
3229        **ds:Signature** element (the *CPA* or *CPP* document). The **ds:Reference** element for the *CPP* or  
3230        *CPA* document can include an IMPLIED **type** attribute that has a value of:

3231        "http://www.w3.org/2000/09/xmldsig#Object"

3232     in accordance with [XMLDSIG]. This attribute is purely informative. It MAY be omitted.  
3233     Implementations of software designed to author or process an ebXML *CPA* or *CPP* document  
3234        SHALL be prepared to handle either case. The **ds:Reference** element can include the **id** attribute,  
3235        type ID, by which this **ds:Reference** element is referenced from a **ds:Signature** element.

3236     **9.9.1.6 ds:Transform element**

3237     The **ds:Reference** element for the *CPA* or *CPP* document SHALL include a descendant  
3238        **ds:Transform** element that excludes the containing **ds:Signature** element and all its descendants.  
3239     This exclusion is achieved by means of specifying the **ds:Algorithm** attribute of the **Transform**  
3240        element as

3241        "http://www.w3.org/2000/09/xmldsig#enveloped-signature"

3242     For example:

```
3253     <ds:Reference ds:URI="">
3254         <ds:Transforms>
3255             <ds:Transform
3256                 ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
3257             </ds:Transforms>
3258             <ds:DigestMethod
3259                 ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
3260                 <ds:DigestValue>...</ds:DigestValue>
3261             </ds:DigestMethod>
3262         </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

3299 of *certId* in Company A's *CPP* would be:

3300  
3301       <tp:Certificate  
3302        tp:certId="CompanyA\_EncryptionCert"><ds:KeyInfo/></tp:Certificate>  
3303

3304 **9.12 Modifying Parameters of the Process-Specification Document Based on**  
3305 **Information in the CPA**

3306 A *Process-Specification* document contains a number of parameters, expressed as XML  
3307 attributes. An example is the security attributes that are counterparts of the attributes of the *CPA*  
3308 ***BusinessTransactionCharacteristics*** element. The values of these attributes can be considered to  
3309 be default values or recommendations. When a *CPA* is created, the *Parties* might decide to  
3310 accept the recommendations in the *Process-Specification* or they MAY agree on values of these  
3311 parameters that better reflect their needs.

3312  
3313 When a *CPA* is used to configure a run-time system, choices specified in the *CPA* MUST always  
3314 assume precedence over choices specified in the referenced *Process-Specification* document. In  
3315 particular, all choices expressed in a *CPA*'s ***BusinessTransactionCharacteristics*** and ***Packaging***  
3316 elements MUST be implemented as agreed to by the *Parties*. These choices SHALL override  
3317 the default values expressed in the *Process-Specification* document. The process of installing the  
3318 information from the *CPA* and *Process-Specification* document MUST verify that all of the  
3319 resulting choices are mutually consistent and MUST signal an error if they are not.

3320

3321       NOTE: There are several ways of overriding the information in the *Process-*  
3322       *Specification* document by information from the *CPA*. For example:

- 3323
- 3324     • The *CPA* composition tool can create a separate copy of the *Process-Specification*  
3325       document. The tool can then directly modify the *Process-Specification* document  
3326       with information from the *CPA*. An advantage of this method is that the override  
3327       process is performed entirely by the *CPA* composition tool.
  - 3328     • A *CPA* installation tool can dynamically override parameters in the *Process-*  
3329       *Specification* document using information from the corresponding parameters in the  
3330       *CPA* at the time the *CPA* and *Process-Specification* document are installed in the  
3331       *Parties'* systems. This eliminates the need to create a separate copy of the *Process-*  
3332       *Specification* document.
  - 3333     • Other possible methods might be based on XSLT transformations of the parameter  
3334       information in the *CPA* and/or the *Process-Specification* document.

## 3335 **10 References**

3336 Some references listed below specify functions for which specific XML definitions are provided  
3337 in the *CPP* and *CPA*. Other specifications are referred to in this specification in the sense that  
3338 they are represented by keywords for which the *Parties* to the *CPA* MAY obtain plug-ins or  
3339 write custom support software but do not require specific XML element sets in the *CPP* and  
3340 *CPA*.

3341  
3342 In a few cases, the only available specification for a function is a proprietary specification.  
3343 These are indicated by notes within the citations below.

3344  
3345 [ccOVER] ebXML Core Components Overview, <http://www.ebxml.org/specs/ccOVER.pdf>.

3346  
3347 [DIGENV] Digital Envelope, RSA Laboratories, <http://www.rsasecurity.com/rsalabs/faq/2-2-4.html>.  
3348 NOTE: At this time, the only available specification for digital envelope appears to be the RSA  
3349 Laboratories specification.

3350  
3351 [ebBPSS] ebXML Business Process Specification Schema, <http://www.ebxml.org/specs/ebBPSS.pdf>.

3352  
3353 [ebMS] ebXML Message Service Specification, [http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS\\_v2\\_0.pdf](http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf).

3355  
3356 [ebRS] ebXML Registry Services Specification, <http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf>.

3358  
3359 [HTTP] Hypertext Transfer Protocol, Internet Engineering Task Force RFC 2616, <http://www.rfc-editor.org/rfc/rfc2616.txt>.

3361  
3362 [IPSEC] IP Security Document Roadmap, Internet Engineering Task Force RFC 2411,  
<http://www.ietf.org/rfc/rfc2411.txt>.

3364  
3365 [ISO6523] Structure for the Identification of Organizations and Organization Parts, International  
3366 Standards Organization ISO-6523.

3367  
3368 [MIME] MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying  
3369 and Describing the Format of Internet *Message* Bodies. Internet Engineering Task Force RFC  
3370 1521, <http://www.ietf.org/rfc/rfc1521.txt>.

3371  
3372 [RFC959] File Transfer Protocol (FTP), Internet Engineering Task Force RFC 959,  
<http://www.ietf.org/rfc/rfc959.txt>.

3374  
3375 [RFC1123] Requirements for Internet Hosts -- Application and Support, Internet Engineering  
3376 Task Force RFC 1123, <http://www.ietf.org/rfc/rfc1123.txt>.

3377  
3378 [RFC1579] Firewall-Friendly FTP, Internet Engineering Task Force RFC 1579,

3379 [RFC2015] MIME Security with Pretty Good PrivacyInternet Engineering Task Force, RFC  
3380 2015, <http://www.ietf.org/rfc/rfc2015.txt>.

3381 [RFC2119] Key Words for use in RFCs to Indicate Requirement Levels, Internet Engineering  
3382 Task Force RFC 2119, <http://www.ietf.org/rfc/rfc2119.txt>.

3383 [RFC2246] The TLS Protocol, Internet Engineering Task Force RFC 2246,  
3384 <http://www.ietf.org/rfc/rfc2246.txt>.

3385 [RFC2251] Lightweight Directory Access Protocol (v3), , Internet Engineering Task Force RFC  
3386 2251, <http://www.ietf.org/rfc/rfc2251.txt>.

3387 [RFC2396] Uniform Resource Identifiers (URI): Generic Syntax, Internet Engineering Task  
3388 Force RFC 2396, <http://www.ietf.org/rfc/rfc2396.txt>.

3389 [RFC2617] HTTP Authentication: Basic and Digest Authentication, , Internet Engineering Task  
3390 Force RFC 2617, <http://www.ietf.org/rfc/rfc2617.txt>.

3391 [RFC2822] Internet Message Format, Internet Engineering Task Force RFC 2822,  
3392 <http://www.ietf.org/rfc/rfc2822.txt>.

3393 [S/MIME] S/MIME Version 3 Message Specification, Internet Engineering Task Force RFC  
3394 2633, <http://www.ietf.org/rfc/rfc2633.txt>.

3395 [SAML] Security Assertion Markup Language, [http://www.oasis-open.org/committees/security/-\\_documents](http://www.oasis-open.org/committees/security/-_documents).

3396 [SMTP] Simple Mail Transfer Protocol, Internet Engineering Task Force RFC 2821,  
3397 <http://www.faqs.org/rfcs/rfc2821.html>.

3398 [SSL] Secure Sockets Layer, Netscape Communications Corp., <http://www.netscape.com/eng/ssl3/>  
3399 NOTE: At this time, it appears that the Netscape specification is the only available specification  
3400 of SSL.

3401 [X12] ANSI X12 Standard for Electronic Data Interchange, X12 Standard Release  
3402 4050, December 2001.

3403 [XAML] Transaction Authority Markup Language, <http://xaml.org/>.

3404 [XLINK] XML Linking Language, <http://www.w3.org/TR/xlink/>.

3405 [XML] Extensible Markup Language (XML), World Wide Web Consortium,  
3406 <http://www.w3.org/XML>.

- 3425 [XMLC14N] Canonical XML, Ver. 1.0, Worldwide Web Consortium,  
3426 <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>.  
3427  
3428 [XMLDSIG] XML Signature Syntax and Processing, Worldwide Web Consortium,  
3429 <http://www.w3.org/TR/xmldsig-core/>.  
3430  
3431 [XMLENC] XML Encryption Syntax and Processing, Worldwide Web Consortium,  
3432 <http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/>.  
3433  
3434 [XMLNS] Namespaces in XML, Worldwide Web Consortium, <http://www.w3.org/TR/REC-xml-names/>.  
3435  
3436  
3437 [XMLSCHEMA-1] XML Schema Part 1: Structures, Worldwide Web Consortium,  
3438 <http://www.w3.org/TR/xmlschema-1/>.  
3439  
3440 [XMLSCHEMA-2] XML Schema Part 2: Datatypes, Worldwide Web Consortium,  
3441 <http://www.w3.org/TR/xmlschema-2/>.  
3442  
3443 [XPOINTER] XML Pointer Language, Worldwide Web Consortium, <http://www.w3.org/TR/xptr/>.  
3444  
3445 [ZLIB] Zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library,  
3446 <http://www.gzip.org/zlib/>.

**3447 11 Conformance**

3448 In order to conform to this specification, an implementation:  
3449   a) SHALL support all the functional and interface requirements defined in this specification,  
3450   b) SHALL NOT specify any requirements that would contradict or cause non-conformance  
3451       to this specification.

3452  
3453 A conforming implementation SHALL satisfy the conformance requirements of the applicable  
3454 parts of this specification.

3455  
3456 An implementation of a tool or service that creates or maintains ebXML *CPP* or *CPA* instance  
3457 documents SHALL be determined to be conformant by validation of the *CPP* or *CPA* instance  
3458 documents, created or modified by said tool or service, against the XML  
3459 Schema[XMLSCHEMA-1] definition of the *CPP* or *CPA* in Appendix D and available from

3460  
3461       [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)  
3462

3463 by using two or more validating XML Schema parsers that conform to the W3C XML Schema  
3464 specifications[XMLSCHEMA-1, XMLSCHEMA-2].

3465  
3466 The objective of conformance testing is to determine whether an implementation being tested  
3467 conforms to the requirements stated in this specification. Conformance testing enables vendors to  
3468 implement compatible and interoperable systems. Implementations and applications SHALL be  
3469 tested using available test suites to verify their conformance to this specification.

3470  
3471 Publicly available test suites from vendor neutral organizations such as OASIS and the U.S.A.  
3472 National Institute of Science and Technology (NIST) SHOULD be used to verify the  
3473 conformance of implementations, applications, and components claiming conformance to this  
3474 specification. Open-source reference implementations might be available to allow vendors to test  
3475 their products for interface compatibility, conformance, and interoperability.  
3476

**3477 12 Disclaimer**

3478 The views and specification expressed in this document are those of the authors and are not  
3479 necessarily those of their employers. The authors and their employers specifically disclaim  
3480 responsibility for any problems arising from correct or incorrect implementation or use of this  
3481 design.

## 3482    **13 Contact Information**

3483

3484    Arvola Chan (Author)

3485    TIBCO Software

3486    3303 Hillview Avenue

3487    Palo Alto, CA 94304

3488    USA

3489    Phone: 650-846-5046

3490    email: <mailto:arvola@tibco.com>

3491

3492    Dale W. Moberg (Author)

3493    Cyclone Commerce

3494    8388 E. Hartford Drive

3495    Scottsdale, AZ 85255

3496    USA

3497    Phone: 480-627-2648

3498    email: <mailto:dmoberg@cyclonecommerce.com>

3499

3500    Himagiri Mukkamala (Author)

3501    Sybase Inc.

3502    5000 Hacienda Dr

3503    Dublin, CA, 84568

3504    USA

3505    Phone: 925-236-5477

3506    email: <mailto:himagiri@sybase.com>

3507

3508    Peter M. Ogden (Author)

3509    Cyclone Commerce, Inc.

3510    8388 East Hartford Drive

3511    Scottsdale, AZ 85255

3512    USA

3513    Phone: 480-627-1800

3514    email: <mailto:pogden@cyclonecommerce.com>

3515

3516    Martin W. Sachs (Author)

3517    IBM T. J. Watson Research Center

3518    P.O.B. 704

3519    Yorktown Hts, NY 10598

3520    USA

3521    Phone: 914-784-7287

3522    email: <mailto:mwsachs@us.ibm.com>

3523

3524    Tony Weida (Coordinating Editor)

3525    535 West 110<sup>th</sup> St., #4J

3526      New York, NY 10025  
3527      USA  
3528      Phone: 212-678-5265  
3529      email: <mailto:rweida@hotmail.com>  
3530  
3531      Jean Zheng  
3532      Vitria  
3533      945 Stewart Drive  
3534      Sunnyvale, CA 94086  
3535      USA  
3536      Phone: 408-212-2468  
3537      email: <mailto:jzheng@vitria.com>

3538

## Notices

3539

3540 Portions of this document are copyright (c) 2001 OASIS and UN/CEFACT.

3541

**Copyright (C) The Organization for the Advancement of Structured Information Standards [OASIS] 2002. All Rights Reserved.**

3544

3545 This document and translations of it may be copied and furnished to others, and derivative works  
3546 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
3547 published and distributed, in whole or in part, without restriction of any kind, provided that the  
3548 above copyright notice and this paragraph are included on all such copies and derivative works.  
3549 However, this document itself may not be modified in any way, such as by removing the  
3550 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
3551 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
3552 Property Rights document must be followed, or as required to translate it into languages other  
3553 than English.

3554

3555 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
3556 successors or assigns.

3557

3558 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
3559 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT  
3560 LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN  
3561 WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF  
3562 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

3563

3564 OASIS takes no position regarding the validity or scope of any intellectual property or other  
3565 rights that might be claimed to pertain to the implementation or use of the technology described  
3566 in this document or the extent to which any license under such rights might or might not be  
3567 available; neither does it represent that it has made any effort to identify any such rights.  
3568 Information on OASIS's procedures with respect to rights in OASIS specifications can be found  
3569 at the OASIS website. Copies of claims of rights made available for publication and any  
3570 assurances of licenses to be made available, or the result of an attempt made to obtain a general  
3571 license or permission for the use of such proprietary rights by implementors or users of this  
3572 specification, can be obtained from the OASIS Executive Director.

3573

3574 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
3575 applications, or other proprietary rights which may cover technology that may be required to  
3576 implement this specification. Please address the information to the OASIS Executive Director.

3577

3578 OASIS has been notified of intellectual property rights claimed in regard to some or all of the  
3579 contents of this specification. For more information consult the online list of claimed rights.

## 3580 Appendix A Example of CPP Document (Non-Normative)

3581 This example includes two CPPs that are used to form the CPA in Appendix B. They are  
3582 available as ASCII files at

3583 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpp-example-companyA-017.xml>  
3584 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpp-example-companyB-017.xml>

3585  
3586 **draft-cpp-example-companyA-017.xml:**  
3587  
3588 <?xml version="1.0"?>  
3589 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->  
3590 <tp:CollaborationProtocolProfile  
3591   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2\_0.xsd"  
3592   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
3593   xmlns:xlink="http://www.w3.org/1999/xlink"  
3594   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
3595   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
3596   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2\_0.xsd  
3597                    draft-cpp-cpa-017.xsd"  
3598   tp:cppid="uri:companyA-cpp" tp:version="017">  
3599   <!-- Party info for CompanyA-->  
3600   <tp:PartyInfo  
3601     tp:partyName="CompanyA"  
3602     tp:defaultMshChannelId="asyncChannelA1"  
3603     tp:defaultMshPackageId="CompanyA\_MshSignalPackage">  
3604     <tp:PartyId  
3605       tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">123456789</tp:PartyId>  
3606     <tp:PartyRef xlink:href="http://CompanyA.com/about.html"/>  
3607     <tp:CollaborationRole  
3608       <tp:ProcessSpecification  
3609         tp:version="2.0"  
3610         tp:name="PIP3A4RequestPurchaseOrder"  
3611         xlink:type="simple"  
3612         xlink:href="http://www.rosettanet.org/processes/3A4.xml"  
3613         tp:uuid="urn:icann:rosettanet.org:bpid:3A4\$2.0"/>  
3614       <tp:Role  
3615         tp:name="Buyer"  
3616         xlink:type="simple"  
3617         xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>  
3618       <tp:ApplicationCertificateRef tp:certId="CompanyA\_AppCert"/>  
3619       <tp:ServiceBinding  
3620         <tp:Service>bpid:icann:rosettanet.org:3A4\$2.0</tp:Service>  
3621         <tp:CanSend  
3622           <tp:ThisPartyActionBinding  
3623             tp:id="companyA\_ABID1"  
3624             tp:action="Purchase Order Request Action"  
3625             tp:packageId="CompanyA\_RequestPackage">  
3626             <tp:BusinessTransactionCharacteristics  
3627               tp:isNonRepudiationRequired="true"  
3628               tp:isNonRepudiationReceiptRequired="true"  
3629               tp:isSecureTransportRequired="true"  
3630               tp:isConfidential="transient"  
3631               tp:isAuthenticated="persistent"  
3632               tp:isTamperProof="persistent"  
3633               tp:isAuthorizationRequired="true"  
3634               tp:timeToAcknowledgeReceipt="PT2H" tp:timeToPerform="P1D"/>  
3635           <tp:ActionContext  
3636             tp:binaryCollaboration="Request Purchase Order"  
3637             tp:businessTransactionActivity="Request Purchase Order"  
3638             tp:requestOrResponseAction="Purchase Order Request Action"/>  
3639           <tp:ChannelId>asyncChannelA1</tp:ChannelId>  
3640           </tp:ThisPartyActionBinding>  
3641       </tp:CanSend>  
3642       <tp:CanSend  
3643         <tp:ThisPartyActionBinding  
3644           tp:id="companyA\_ABID2"

```
3645      tp:action="ReceiptAcknowledgement"
3646      tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
3647      <tp:BusinessTransactionCharacteristics
3648          tp:isNonRepudiationRequired="true"
3649          tp:isNonRepudiationReceiptRequired="true"
3650          tp:isSecureTransportRequired="true"
3651          tp:isConfidential="transient"
3652          tp:isAuthenticated="persistent"
3653          tp:isTamperProof="persistent"
3654          tp:isAuthorizationRequired="true"
3655          tp:timeToAcknowledgeReceipt="PT2H"
3656          tp:timeToPerform="P1D"/>
3657      <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3658  </tp:ThisPartyActionBinding>
3659  </tp:CanSend>
3660  <!-- The next binding uses a synchronous delivery channel -->
3661  <tp:CanSend>
3662      <tp:ThisPartyActionBinding
3663          tp:id="companyA_ABID6"
3664          tp:action="Purchase Order Request Action"
3665          tp:packageId="CompanyA_RequestPackage">
3666          <tp:BusinessTransactionCharacteristics
3667              tp:isNonRepudiationRequired="true"
3668              tp:isNonRepudiationReceiptRequired="true"
3669              tp:isSecureTransportRequired="true"
3670              tp:isConfidential="transient"
3671              tp:isAuthenticated="persistent"
3672              tp:isTamperProof="persistent"
3673              tp:isAuthorizationRequired="true"
3674              tp:timeToAcknowledgeReceipt="PT5M"
3675              tp:timeToPerform="PT5M"/>
3676          <tp:ActionContext
3677              tp:binaryCollaboration="Request Purchase Order"
3678              tp:businessTransactionActivity="Request Purchase Order"
3679              tp:requestOrResponseAction="Purchase Order Request Action"/>
3680          <tp:ChannelId>syncChannelA1</tp:ChannelId>
3681  </tp:ThisPartyActionBinding>
3682  <tp:CanReceive>
3683      <tp:ThisPartyActionBinding
3684          tp:id="companyA_ABID7"
3685          tp:action="Purchase Order Confirmation Action"
3686          tp:packageId="CompanyA_SyncReplyPackage">
3687          <tp:BusinessTransactionCharacteristics
3688              tp:isNonRepudiationRequired="true"
3689              tp:isNonRepudiationReceiptRequired="true"
3690              tp:isSecureTransportRequired="true"
3691              tp:isConfidential="transient"
3692              tp:isAuthenticated="persistent"
3693              tp:isTamperProof="persistent"
3694              tp:isAuthorizationRequired="true"
3695              tp:timeToAcknowledgeReceipt="PT5M"
3696              tp:timeToPerform="PT5M"/>
3697          <tp:ActionContext
3698              tp:binaryCollaboration="Request Purchase Order"
3699              tp:businessTransactionActivity="Request Purchase Order"
3700              tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
3701          <tp:ChannelId>syncChannelA1</tp:ChannelId>
3702      </tp:ThisPartyActionBinding>
3703  </tp:CanReceive>
3704  <tp:CanReceive>
3705      <tp:ThisPartyActionBinding
3706          tp:id="companyA_ABID8"
3707          tp:action="Exception"
3708          tp:packageId="CompanyA_ExceptionPackage">
3709          <tp:BusinessTransactionCharacteristics
3710              tp:isNonRepudiationRequired="true"
3711              tp:isNonRepudiationReceiptRequired="true"
3712              tp:isSecureTransportRequired="true"
3713              tp:isConfidential="transient"
3714              tp:isAuthenticated="persistent"
3715              tp:isTamperProof="persistent"
```

```
3716          tp:isAuthorizationRequired="true"
3717          tp:timeToAcknowledgeReceipt="PT5M"
3718          tp:timeToPerform="PT5M"/>
3719      <tp:ChannelId>syncChannelA1</tp:ChannelId>
3720    </tp:ThisPartyActionBinding>
3721  </tp:CanReceive>
3722 </tp:CanSend>
3723 <tp:CanReceive>
3724   <tp:ThisPartyActionBinding
3725     tp:id="companyA_ABID3"
3726     tp:action="Purchase Order Confirmation Action"
3727     tp:packageId="CompanyA_ResponsePackage">
3728     <tp:BusinessTransactionCharacteristics
3729       tp:isNonRepudiationRequired="true"
3730       tp:isNonRepudiationReceiptRequired="true"
3731       tp:isSecureTransportRequired="true"
3732       tp:isConfidential="transient"
3733       tp:isAuthenticated="persistent"
3734       tp:isTamperProof="persistent"
3735       tp:isAuthorizationRequired="true"
3736       tp:timeToAcknowledgeReceipt="PT2H"
3737       tp:timeToPerform="P1D"/>
3738   <tp:ActionContext
3739     tp:binaryCollaboration="Request Purchase Order"
3740     tp:businessTransactionActivity="Request Purchase Order"
3741     tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
3742   <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3743 </tp:ThisPartyActionBinding>
3744 </tp:CanReceive>
3745 <tp:CanReceive>
3746   <tp:ThisPartyActionBinding
3747     tp:id="companyA_ABID4"
3748     tp:action="ReceiptAcknowledgment"
3749     tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
3750     <tp:BusinessTransactionCharacteristics
3751       tp:isNonRepudiationRequired="true"
3752       tp:isNonRepudiationReceiptRequired="true"
3753       tp:isSecureTransportRequired="true"
3754       tp:isConfidential="transient"
3755       tp:isAuthenticated="persistent" tp:isTamperProof="persistent"
3756       tp:isAuthorizationRequired="true"
3757       tp:timeToAcknowledgeReceipt="PT2H"
3758       tp:timeToPerform="P1D"/>
3759     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3760   </tp:ThisPartyActionBinding>
3761 </tp:CanReceive>
3762 <tp:CanReceive>
3763   <tp:ThisPartyActionBinding
3764     tp:id="companyA_ABID5"
3765     tp:action="Exception"
3766     tp:packageId="CompanyA_ExceptionPackage">
3767     <tp:BusinessTransactionCharacteristics
3768       tp:isNonRepudiationRequired="true"
3769       tp:isNonRepudiationReceiptRequired="true"
3770       tp:isSecureTransportRequired="true"
3771       tp:isConfidential="transient"
3772       tp:isAuthenticated="persistent"
3773       tp:isTamperProof="persistent"
3774       tp:isAuthorizationRequired="true"
3775       tp:timeToAcknowledgeReceipt="PT2H"
3776       tp:timeToPerform="P1D"/>
3777     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3778   </tp:ThisPartyActionBinding>
3779 </tp:CanReceive>
3780 </tp:ServiceBinding>
3781 </tp:CollaborationRole>
3782 <!-- Certificates used by the "Buyer" company -->
3783 <tp:Certificate tp:certId="CompanyA_AppCert">
3784   <ds:KeyInfo>
3785     <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
3786   </ds:KeyInfo>
```

```
3787     </tp:Certificate>
3788     <tp:Certificate tp:certId="CompanyA_SigningCert">
3789         <ds:KeyInfo>
3790             <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
3791         </ds:KeyInfo>
3792     </tp:Certificate>
3793     <tp:Certificate tp:certId="CompanyA_EncryptionCert">
3794         <ds:KeyInfo>
3795             <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
3796         </ds:KeyInfo>
3797     </tp:Certificate>
3798     <tp:Certificate tp:certId="CompanyA_ServerCert">
3799         <ds:KeyInfo>
3800             <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
3801         </ds:KeyInfo>
3802     </tp:Certificate>
3803     <tp:Certificate tp:certId="CompanyA_ClientCert">
3804         <ds:KeyInfo>
3805             <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
3806         </ds:KeyInfo>
3807     </tp:Certificate>
3808     <tp:Certificate tp:certId="TrustedRootCertA1">
3809         <ds:KeyInfo>
3810             <ds:KeyName>TrustedRootCertA1_Key</ds:KeyName>
3811         </ds:KeyInfo>
3812     </tp:Certificate>
3813     <tp:Certificate tp:certId="TrustedRootCertA2">
3814         <ds:KeyInfo>
3815             <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
3816         </ds:KeyInfo>
3817     </tp:Certificate>
3818     <tp:Certificate tp:certId="TrustedRootCertA3">
3819         <ds:KeyInfo>
3820             <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
3821         </ds:KeyInfo>
3822     </tp:Certificate>
3823     <tp:Certificate tp:certId="TrustedRootCertA4">
3824         <ds:KeyInfo>
3825             <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
3826         </ds:KeyInfo>
3827     </tp:Certificate>
3828     <tp:Certificate tp:certId="TrustedRootCertA5">
3829         <ds:KeyInfo>
3830             <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
3831         </ds:KeyInfo>
3832     </tp:Certificate>
3833     <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
3834         <tp:TrustAnchors>
3835             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1"/>
3836             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2"/>
3837             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4"/>
3838         </tp:TrustAnchors>
3839     </tp:SecurityDetails>
3840     <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
3841         <tp:TrustAnchors>
3842             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
3843             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
3844         </tp:TrustAnchors>
3845     </tp:SecurityDetails>
3846     <!-- An asynchronous delivery channel -->
3847     <tp:DeliveryChannel
3848         tp:channelId="asyncChannelA1"
3849         tp:transportId="transportA2"
3850         tp:docExchangeId="docExchangeA1">
3851         <tp:MessagingCharacteristics
3852             tp:syncReplyMode="none"
3853             tp:ackRequested="always"
3854             tp:ackSignatureRequested="always"
3855             tp:duplicateElimination="always"/>
3856     </tp:DeliveryChannel>
3857     <!-- A synchronous delivery channel -->
```

```
3858     <tp:DeliveryChannel
3859         tp:channelId="syncChannelA1"
3860         tp:transportId="transportA1"
3861         tp:docExchangeId="docExchangeA1">
3862         <tp:MessagingCharacteristics
3863             tp:syncReplyMode="none"
3864             tp:ackRequested="always"
3865             tp:ackSignatureRequested="always"
3866             tp:duplicateElimination="always"/>
3867     </tp:DeliveryChannel>
3868     <tp:Transport tp:transportId="transportA1">
3869         <tp:TransportSender>
3870             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3871             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3872             <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3873             <tp:TransportClientSecurity>
3874                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3875                 <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
3876                 <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3877             </tp:TransportClientSecurity>
3878         </tp:TransportSender>
3879         <tp:TransportReceiver>
3880             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3881             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3882             <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3883             <tp:Endpoint
3884                 tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
3885                 tp:type="allPurpose"/>
3886             <tp:TransportServerSecurity>
3887                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3888                 <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
3889                 <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3890             </tp:TransportServerSecurity>
3891         </tp:TransportReceiver>
3892     </tp:Transport>
3893     <tp:Transport tp:transportId="transportA2">
3894         <tp:TransportSender>
3895             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3896             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3897             <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3898             <tp:TransportClientSecurity>
3899                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3900                 <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
3901                 <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3902             </tp:TransportClientSecurity>
3903         </tp:TransportSender>
3904         <tp:TransportReceiver>
3905             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3906             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3907             <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3908             <tp:Endpoint
3909                 tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
3910                 tp:type="allPurpose"/>
3911             <tp:TransportServerSecurity>
3912                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3913                 <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
3914                 <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3915             </tp:TransportServerSecurity>
3916         </tp:TransportReceiver>
3917     </tp:Transport>
3918     <tp:DocExchange tp:docExchangeId="docExchangeA1">
3919         <tp:ebXMLSenderBinding tp:version="2.0">
3920             <tp:ReliableMessaging>
3921                 <tp:Retries>3</tp:Retries>
3922                 <tp:RetryInterval>PT2H</tp:RetryInterval>
3923                 <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
3924             </tp:ReliableMessaging>
3925             <tp:PersistDuration>P1D</tp:PersistDuration>
3926             <tp:SenderNonRepudiation>
3927
3928         <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
```

```
3929      <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
3930      <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
3931      sha1</tp:SignatureAlgorithm>
3932          <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
3933      </tp:SenderNonRepudiation>
3934      <tp:SenderDigitalEnvelope>
3935          <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
3936          <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
3937          <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
3938      </tp:SenderDigitalEnvelope>
3939  </tp:ebXMLSenderBinding>
3940  <tp:ebXMLReceiverBinding tp:version="2.0">
3941      <tp:ReliableMessaging>
3942          <tp:Retries>3</tp:Retries>
3943          <tp:RetryInterval>PT2H</tp:RetryInterval>
3944          <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
3945      </tp:ReliableMessaging>
3946      <tp:PersistDuration>P1D</tp:PersistDuration>
3947      <tp:ReceiverNonRepudiation>
3948
3949  <tp:NonRepudiationProtocol>></tp:NonRepudiationProtocol>
3950      <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
3951      <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
3953      sha1</tp:SignatureAlgorithm>
3954          <tp:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
3955      </tp:ReceiverNonRepudiation>
3956      <tp:ReceiverDigitalEnvelope>
3957          <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
3958          <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
3959          <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
3960      </tp:ReceiverDigitalEnvelope>
3961  </tp:ebXMLReceiverBinding>
3962  </tp:DocExchange>
3963  </tp:PartyInfo>
3964  <!-- SimplePart corresponding to the SOAP Envelope -->
3965  <tp:SimplePart
3966      tp:id="CompanyA_MsgHdr"
3967      tp:mimetype="text/xml">
3968      <tp:NamespaceSupported
3969          tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2\_0.xsd"
3970          tp:version="2.0">
3971          http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2\_0.xsd
3972      </tp:NamespaceSupported>
3973  </tp:SimplePart>
3974  <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
3975  <tp:SimplePart
3976      tp:id="CompanyA_ReceiptAcknowledgment"
3977      tp:mimetype="application/xml">
3978      <tp:NamespaceSupported
3979          tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
3980          tp:version="2.0">
3981          http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
3982      </tp:NamespaceSupported>
3983  </tp:SimplePart>
3984  <!-- SimplePart corresponding to an Exception business signal -->
3985  <tp:SimplePart
3986      tp:id="CompanyA_Exception"
3987      tp:mimetype="application/xml">
3988      <tp:NamespaceSupported
3989          tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2\_0.xsd"
3990          tp:version="2.0">
3991          http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2\_0.xsd
3992      </tp:NamespaceSupported>
3993  </tp:SimplePart>
3994  <!-- SimplePart corresponding to a request action -->
3995  <tp:SimplePart
3996      tp:id="CompanyA_Request"
3997      tp:mimetype="application/xml">
3998      <tp:NamespaceSupported
3999          tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
          tp:version="2.0">
```

```
4000      http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
4001      </tp:NamespaceSupported>
4002  </tp:SimplePart>
4003  <!-- SimplePart corresponding to a response action -->
4004  <tp:SimplePart
4005    tp:id="CompanyA_Response"
4006    tp:mimetype="application/xml">
4007    <tp:NamespaceSupported
4008      tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
4009      tp:version="2.0">
4010    http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
4011  </tp:NamespaceSupported>
4012  </tp:SimplePart>
4013  <!-- An ebXML message with a SOAP Envelope only -->
4014  <tp:Packaging tp:id="CompanyA_MshSignalPackage">
4015    <tp:ProcessingCapabilities
4016      tp:parse="true"
4017      tp:generate="true"/>
4018    <tp:CompositeList>
4019      <tp:Composite
4020        tp:id="CompanyA_MshSignal"
4021        tp:mimetype="multipart/related"
4022        tp:mimeparameters="type=text/xml">
4023          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4024        </tp:Composite>
4025      </tp:CompositeList>
4026    </tp:Packaging>
4027  <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
4028  <tp:Packaging tp:id="CompanyA_RequestPackage">
4029    <tp:ProcessingCapabilities
4030      tp:parse="true"
4031      tp:generate="true"/>
4032    <tp:CompositeList>
4033      <tp:Composite
4034        tp:id="CompanyA_RequestMsg"
4035        tp:mimetype="multipart/related"
4036        tp:mimeparameters="type=text/xml">
4037          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4038          <tp:Constituent tp:idref="CompanyA_Request"/>
4039        </tp:Composite>
4040      </tp:CompositeList>
4041    </tp:Packaging>
4042  <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
4043  <tp:Packaging tp:id="CompanyA_ResponsePackage">
4044    <tp:ProcessingCapabilities
4045      tp:parse="true"
4046      tp:generate="true"/>
4047    <tp:CompositeList>
4048      <tp:Composite
4049        tp:id="CompanyA_ResponseMsg"
4050        tp:mimetype="multipart/related"
4051        tp:mimeparameters="type=text/xml">
4052          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4053          <tp:Constituent tp:idref="CompanyA_Response"/>
4054        </tp:Composite>
4055      </tp:CompositeList>
4056    </tp:Packaging>
4057  <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
4058      or an ebXML message with an Exception signal -->
4059  <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
4060    <tp:ProcessingCapabilities
4061      tp:parse="true"
4062      tp:generate="true"/>
4063    <tp:CompositeList>
4064      <tp:Composite
4065        tp:id="CompanyA_SignalAndResponseMsg"
4066        tp:mimetype="multipart/related"
4067        tp:mimeparameters="type=text/xml">
4068          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4069          <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
4070          <tp:Constituent tp:idref="CompanyA_Response"/>
```

```

4071      </tp:Composite>
4072      </tp:CompositeList>
4073  </tp:Packaging>
4074  <!-- An ebXML message with a SOAP Envelope plus a ReceiptAcknowledgment payload -->
4075  <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
4076    <tp:ProcessingCapabilities
4077      tp:parse="true"
4078      tp:generate="true"/>
4079    <tp:CompositeList>
4080      <tp:Composite
4081        tp:id="CompanyA_ReceiptAcknowledgmentMsg"
4082        tp:mimetype="multipart/related"
4083        tp:mimeparameters="type=text/xml">
4084          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4085          <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
4086        </tp:Composite>
4087      </tp:CompositeList>
4088    </tp:Packaging>
4089  <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
4090  <tp:Packaging tp:id="CompanyA_ExceptionPackage">
4091    <tp:ProcessingCapabilities
4092      tp:parse="true"
4093      tp:generate="true"/>
4094    <tp:CompositeList>
4095      <tp:Composite
4096        tp:id="CompanyA_ExceptionMsg"
4097        tp:mimetype="multipart/related"
4098        tp:mimeparameters="type=text/xml">
4099          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4100          <tp:Constituent tp:idref="CompanyA_Exception"/>
4101        </tp:Composite>
4102      </tp:CompositeList>
4103    </tp:Packaging>
4104    <tp:Comment xml:lang="en-US">Buyer's Collaboration Protocol Profile</tp:Comment>
4105  </tp:CollaborationProtocolProfile>
4106
4107
4108 draft-cpp-example-companyB-017.xml:
4109
4110 <?xml version="1.0"?>
4111 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
4112 <tp:CollaborationProtocolProfile
4113   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
4114   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4115   xmlns:xlink="http://www.w3.org/1999/xlink"
4116   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4117   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4118   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
4119           draft-cpp-cpa-017.xsd"
4120   tp:cppid="uri:companyB-cpp"
4121   tp:version="017">
4122   <!-- Party info for CompanyB-->
4123   <tp:PartyInfo
4124     tp:partyName="CompanyB"
4125     tp:defaultMshChannelId="asyncChannelB1"
4126     tp:defaultMshPackageId="CompanyB_MshSignalPackage">
4127     <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
4128     <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html"/>
4129     <tp:CollaborationRole>
4130       <tp:ProcessSpecification
4131         tp:version="2.0"
4132         tp:name="PIP3A4RequestPurchaseOrder"
4133         xlink:type="simple" xlink:href="http://www.rosettanet.org/processes/3A4.xml"
4134         tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
4135       <tp:Role
4136         tp:name="Seller"
4137         xlink:type="simple"
4138         xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
4139       <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert"/>
4140       <tp:ServiceBinding>
4141         <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>

```

```
4142      <tp:CanSend>
4143          <tp:ThisPartyActionBinding
4144              tp:id="companyB_ABID1"
4145              tp:action="Purchase Order Confirmation Action"
4146              tp:packageId="CompanyB_ResponsePackage">
4147                  <tp:BusinessTransactionCharacteristics
4148                      tp:isNonRepudiationRequired="true"
4149                      tp:isNonRepudiationReceiptRequired="true"
4150                      tp:isSecureTransportRequired="true"
4151                      tp:isConfidential="transient"
4152                      tp:isAuthenticated="persistent"
4153                      tp:isTamperProof="persistent"
4154                      tp:isAuthorizationRequired="true"
4155                      tp:timeToAcknowledgeReceipt="PT2H"
4156                      tp:timeToPerform="P1D"/>
4157                  <tp:ActionContext
4158                      tp:binaryCollaboration="Request Purchase Order"
4159                      tp:businessTransactionActivity="Request Purchase Order"
4160                      tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4161                  <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4162          </tp:ThisPartyActionBinding>
4163      </tp:CanSend>
4164      <tp:CanSend>
4165          <tp:ThisPartyActionBinding
4166              tp:id="companyB_ABID2"
4167              tp:action="ReceiptAcknowledgement"
4168              tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
4169                  <tp:BusinessTransactionCharacteristics
4170                      tp:isNonRepudiationRequired="true"
4171                      tp:isNonRepudiationReceiptRequired="true"
4172                      tp:isSecureTransportRequired="true"
4173                      tp:isConfidential="transient"
4174                      tp:isAuthenticated="persistent"
4175                      tp:isTamperProof="persistent"
4176                      tp:isAuthorizationRequired="true"
4177                      tp:timeToAcknowledgeReceipt="PT2H"
4178                      tp:timeToPerform="P1D"/>
4179                  <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4180          </tp:ThisPartyActionBinding>
4181      </tp:CanSend>
4182      <tp:CanSend>
4183          <tp:ThisPartyActionBinding
4184              tp:id="companyB_ABID3"
4185              tp:action="Exception"
4186              tp:packageId="CompanyB_ExceptionPackage">
4187                  <tp:BusinessTransactionCharacteristics
4188                      tp:isNonRepudiationRequired="true"
4189                      tp:isNonRepudiationReceiptRequired="true"
4190                      tp:isSecureTransportRequired="true"
4191                      tp:isConfidential="transient"
4192                      tp:isAuthenticated="persistent"
4193                      tp:isTamperProof="persistent"
4194                      tp:isAuthorizationRequired="true"
4195                      tp:timeToAcknowledgeReceipt="PT2H"
4196                      tp:timeToPerform="P1D"/>
4197                  <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4198          </tp:ThisPartyActionBinding>
4199      </tp:CanSend>
4200      <tp:CanReceive>
4201          <tp:ThisPartyActionBinding
4202              tp:id="companyB_ABID4"
4203              tp:action="Purchase Order Request Action"
4204              tp:packageId="CompanyB_RequestPackage">
4205                  <tp:BusinessTransactionCharacteristics
4206                      tp:isNonRepudiationRequired="true"
4207                      tp:isNonRepudiationReceiptRequired="true"
4208                      tp:isSecureTransportRequired="true"
4209                      tp:isConfidential="transient"
4210                      tp:isAuthenticated="persistent"
4211                      tp:isTamperProof="persistent"
4212                      tp:isAuthorizationRequired="true"
```

```
4213         tp:timeToAcknowledgeReceipt="PT2H"
4214         tp:timeToPerform="P1D"/>
4215     <tp:ActionContext
4216         tp:binaryCollaboration="Request Purchase Order"
4217         tp:businessTransactionActivity="Request Purchase Order"
4218         tp:requestOrResponseAction="Purchase Order Request Action"/>
4219         <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4220     </tp:ThisPartyActionBinding>
4221 </tp:CanReceive>
4222 <tp:CanReceive>
4223     <tp:ThisPartyActionBinding
4224         tp:id="companyB_ABID5" tp:action="ReceiptAcknowledgment"
4225         tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
4226         <tp:BusinessTransactionCharacteristics
4227             tp:isNonRepudiationRequired="true"
4228             tp:isNonRepudiationReceiptRequired="true"
4229             tp:isSecureTransportRequired="true"
4230             tp:isConfidential="transient"
4231             tp:isAuthenticated="persistent"
4232             tp:isTamperProof="persistent"
4233             tp:isAuthorizationRequired="true"
4234             tp:timeToAcknowledgeReceipt="PT2H"
4235             tp:timeToPerform="P1D"/>
4236         <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4237     </tp:ThisPartyActionBinding>
4238 </tp:CanReceive>
4239 <!-- The next binding uses a synchronous delivery channel -->
4240 <tp:CanReceive>
4241     <tp:ThisPartyActionBinding
4242         tp:id="companyB_ABID8"
4243         tp:action="Purchase Order Request Action"
4244         tp:packageId="CompanyB_SyncReplyPackage">
4245         <tp:BusinessTransactionCharacteristics
4246             tp:isNonRepudiationRequired="true"
4247             tp:isNonRepudiationReceiptRequired="true"
4248             tp:isSecureTransportRequired="true"
4249             tp:isConfidential="transient"
4250             tp:isAuthenticated="persistent"
4251             tp:isTamperProof="persistent"
4252             tp:isAuthorizationRequired="true"
4253             tp:timeToAcknowledgeReceipt="PT5M"
4254             tp:timeToPerform="PT5M"/>
4255     <tp:ActionContext
4256         tp:binaryCollaboration="Request Purchase Order"
4257         tp:businessTransactionActivity="Request Purchase Order"
4258         tp:requestOrResponseAction="Purchase Order Request Action"/>
4259         <tp:ChannelId>syncChannelB1</tp:ChannelId>
4260     </tp:ThisPartyActionBinding>
4261 <tp:CanSend>
4262     <tp:ThisPartyActionBinding
4263         tp:id="companyB_ABID6"
4264         tp:action="Purchase Order Confirmation Action"
4265         tp:packageId="CompanyB_ResponsePackage">
4266         <tp:BusinessTransactionCharacteristics
4267             tp:isNonRepudiationRequired="true"
4268             tp:isNonRepudiationReceiptRequired="true"
4269             tp:isSecureTransportRequired="true"
4270             tp:isConfidential="transient"
4271             tp:isAuthenticated="persistent"
4272             tp:isTamperProof="persistent"
4273             tp:isAuthorizationRequired="true"
4274             tp:timeToAcknowledgeReceipt="PT5M"
4275             tp:timeToPerform="PT5M"/>
4276     <tp:ActionContext
4277         tp:binaryCollaboration="Request Purchase Order"
4278         tp:businessTransactionActivity="Request Purchase Order"
4279         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4280         <tp:ChannelId>syncChannelB1</tp:ChannelId>
4281     </tp:ThisPartyActionBinding>
4282 </tp:CanSend>
4283 <tp:CanSend>
```

```
4284      <tp:ThisPartyActionBinding
4285          tp:id="companyB_ABID7"
4286          tp:action="Exception"
4287          tp:packageId="CompanyB_ExceptionPackage">
4288          <tp:BusinessTransactionCharacteristics
4289              tp:isNonRepudiationRequired="true"
4290              tp:isNonRepudiationReceiptRequired="true"
4291              tp:isSecureTransportRequired="true"
4292              tp:isConfidential="transient"
4293              tp:isAuthenticated="persistent"
4294              tp:isTamperProof="persistent"
4295              tp:isAuthorizationRequired="true"
4296              tp:timeToAcknowledgeReceipt="PT5M"
4297              tp:timeToPerform="PT5M"/>
4298          <tp:ChannelId>syncChannelB1</tp:ChannelId>
4299          </tp:ThisPartyActionBinding>
4300      </tp:CanSend>
4301      </tp:CanReceive>
4302      </tp:ServiceBinding>
4303  </tp:CollaborationRole>
4304  <!-- Certificates used by the "Seller" company -->
4305  <tp:Certificate tp:certId="CompanyB_AppCert">
4306      <ds:KeyInfo>
4307          <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
4308      </ds:KeyInfo>
4309  </tp:Certificate>
4310  <tp:Certificate tp:certId="CompanyB_SigningCert">
4311      <ds:KeyInfo>
4312          <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
4313      </ds:KeyInfo>
4314  </tp:Certificate>
4315  <tp:Certificate tp:certId="CompanyB_EncryptionCert">
4316      <ds:KeyInfo>
4317          <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
4318      </ds:KeyInfo>
4319  </tp:Certificate>
4320  <tp:Certificate tp:certId="CompanyB_ServerCert">
4321      <ds:KeyInfo>
4322          <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
4323      </ds:KeyInfo>
4324  </tp:Certificate>
4325  <tp:Certificate tp:certId="CompanyB_ClientCert">
4326      <ds:KeyInfo>
4327          <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
4328      </ds:KeyInfo>
4329  </tp:Certificate>
4330  <tp:Certificate tp:certId="TrustedRootCertB4">
4331      <ds:KeyInfo>
4332          <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
4333      </ds:KeyInfo>
4334  </tp:Certificate>
4335  <tp:Certificate tp:certId="TrustedRootCertB5">
4336      <ds:KeyInfo>
4337          <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
4338      </ds:KeyInfo>
4339  </tp:Certificate>
4340  <tp:Certificate tp:certId="TrustedRootCertB6">
4341      <ds:KeyInfo>
4342          <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
4343      </ds:KeyInfo>
4344  </tp:Certificate>
4345  <tp:Certificate tp:certId="TrustedRootCertB7">
4346      <ds:KeyInfo>
4347          <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
4348      </ds:KeyInfo>
4349  </tp:Certificate>
4350  <tp:Certificate tp:certId="TrustedRootCertB8">
4351      <ds:KeyInfo>
4352          <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
4353      </ds:KeyInfo>
4354  </tp:Certificate>
```

```
4355 <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
4356   <tp:TrustAnchors>
4357     <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5"/>
4358     <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6"/>
4359     <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4"/>
4360   </tp:TrustAnchors>
4361 </tp:SecurityDetails>
4362 <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
4363   <tp:TrustAnchors>
4364     <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8"/>
4365     <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7"/>
4366   </tp:TrustAnchors>
4367 </tp:SecurityDetails>
4368 <!-- An asynchronous delivery channel -->
4369 <tp:DeliveryChannel
4370   tp:channelId="asyncChannelB1"
4371   tp:transportId="transportB1"
4372   tp:docExchangeId="docExchangeB1">
4373   <tp:MessagingCharacteristics
4374     tp:syncReplyMode="none"
4375     tp:ackRequested="always"
4376     tp:ackSignatureRequested="always"
4377     tp:duplicateElimination="always"/>
4378 </tp:DeliveryChannel>
4379 <!-- A synchronous delivery channel -->
4380 <tp:DeliveryChannel
4381   tp:channelId="syncChannelB1"
4382   tp:transportId="transportB2"
4383   tp:docExchangeId="docExchangeB1">
4384   <tp:MessagingCharacteristics
4385     tp:syncReplyMode="signalsAndResponse"
4386     tp:ackRequested="always"
4387     tp:ackSignatureRequested="always"
4388     tp:duplicateElimination="always"/>
4389 </tp:DeliveryChannel>
4390 <tp:Transport tp:transportId="transportB1">
4391   <tp:TransportSender
4392     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4393     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4394     <tp:TransportClientSecurity>
4395       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4396       <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
4397       <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4398     </tp:TransportClientSecurity>
4399   </tp:TransportSender>
4400   <tp:TransportReceiver
4401     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4402     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4403     <tp:Endpoint
4404       tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
4405       tp:type="allPurpose"/>
4406     <tp:TransportServerSecurity>
4407       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4408       <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
4409       <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4410     </tp:TransportServerSecurity>
4411   </tp:TransportReceiver>
4412 </tp:Transport>
4413 <tp:Transport tp:transportId="transportB2">
4414   <tp:TransportSender
4415     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4416     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4417     <tp:TransportClientSecurity>
4418       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4419       <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
4420       <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4421     </tp:TransportClientSecurity>
4422   </tp:TransportSender>
4423   <tp:TransportReceiver
4424     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4425     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
```

```
4426      <tp:Endpoint
4427          tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
4428          tp:type="allPurpose"/>
4429      <tp:TransportServerSecurity>
4430          <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4431          <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
4432          <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4433      </tp:TransportServerSecurity>
4434  </tp:TransportReceiver>
4435 </tp:Transport>
4436 <tp:DocExchange tp:docExchangeId="docExchangeB1">
4437     <tp:ebXMLSenderBinding tp:version="2.0">
4438         <tp:ReliableMessaging>
4439             <tp:Retries>3</tp:Retries>
4440             <tp:RetryInterval>PT2H</tp:RetryInterval>
4441             <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4442         </tp:ReliableMessaging>
4443         <tp:PersistDuration>P1D</tp:PersistDuration>
4444         <tp:SenderNonRepudiation>
4445
4446     <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4447         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4448         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4449         sha1</tp:SignatureAlgorithm>
4450             <tp:SigningCertificateRef tp:certId="CompanyB_SigningCert"/>
4451         </tp:SenderNonRepudiation>
4452         <tp:SenderDigitalEnvelope>
4453             <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4454             <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4455             <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
4456         </tp:SenderDigitalEnvelope>
4457     </tp:ebXMLSenderBinding>
4458     <tp:ebXMLReceiverBinding tp:version="2.0">
4459         <tp:ReliableMessaging>
4460             <tp:Retries>3</tp:Retries>
4461             <tp:RetryInterval>PT2H</tp:RetryInterval>
4462             <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4463         </tp:ReliableMessaging>
4464         <tp:PersistDuration>P1D</tp:PersistDuration>
4465         <tp:ReceiverNonRepudiation>
4466
4467     <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4468         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4469         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4470         sha1</tp:SignatureAlgorithm>
4471             <tp:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
4472         </tp:ReceiverNonRepudiation>
4473         <tp:ReceiverDigitalEnvelope>
4474             <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4475             <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4476             <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert"/>
4477         </tp:ReceiverDigitalEnvelope>
4478     </tp:ebXMLReceiverBinding>
4479     </tp:DocExchange>
4480   </tp:PartyInfo>
4481   <!-- SimplePart corresponding to the SOAP Envelope -->
4482   <tp:SimplePart
4483       tp:id="CompanyB_MsgHdr"
4484       tp:mimetype="text/xml">
4485       <tp:NamespaceSupported
4486           tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
4487           tp:version="2.0">
4488           http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
4489       </tp:NamespaceSupported>
4490   </tp:SimplePart>
4491   <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
4492   <tp:SimplePart
4493       tp:id="CompanyB_ReceiptAcknowledgment"
4494       tp:mimetype="application/xml">
4495       <tp:NamespaceSupported
4496           tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
```

```
4497      tp:version="2.0">
4498      http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
4499    </tp:NamespaceSupported>
4500  </tp:SimplePart>
4501  <!-- SimplePart corresponding to an Exception business signal -->
4502  <tp:SimplePart
4503    tp:id="CompanyB_Exception"
4504    tp:mimetype="application/xml">
4505    <tp:NamespaceSupported
4506      tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
4507      tp:version="2.0">
4508      http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
4509    </tp:NamespaceSupported>
4510  </tp:SimplePart>
4511  <!-- SimplePart corresponding to a request action -->
4512  <tp:SimplePart
4513    tp:id="CompanyB_Request"
4514    tp:mimetype="application/xml">
4515    <tp:NamespaceSupported
4516      tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
4517      tp:version="2.0">
4518      http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
4519    </tp:NamespaceSupported>
4520  </tp:SimplePart>
4521  <!-- SimplePart corresponding to a response action -->
4522  <tp:SimplePart
4523    tp:id="CompanyB_Response"
4524    tp:mimetype="application/xml">
4525    <tp:NamespaceSupported
4526      tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd.xsd"
4527      tp:version="2.0">
4528      http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
4529    </tp:NamespaceSupported>
4530  </tp:SimplePart>
4531  <!-- An ebXML message with a SOAP Envelope only -->
4532  <tp:Packaging tp:id="CompanyB_MshSignalPackage">
4533    <tp:ProcessingCapabilities
4534      tp:parse="true"
4535      tp:generate="true"/>
4536    <tp:CompositeList>
4537      <tp:Composite
4538        tp:id="CompanyB_MshSignal"
4539        tp:mimetype="multipart/related"
4540        tp:mimeparameters="type=text/xml">
4541          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4542        </tp:Composite>
4543      </tp:CompositeList>
4544    </tp:Packaging>
4545  <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
4546  <tp:Packaging tp:id="CompanyB_RequestPackage">
4547    <tp:ProcessingCapabilities
4548      tp:parse="true"
4549      tp:generate="true"/>
4550    <tp:CompositeList>
4551      <tp:Composite
4552        tp:id="RequestMsg"
4553        tp:mimetype="multipart/related"
4554        tp:mimeparameters="type=text/xml">
4555          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4556          <tp:Constituent tp:idref="CompanyB_Request"/>
4557        </tp:Composite>
4558      </tp:CompositeList>
4559    </tp:Packaging>
4560  <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
4561  <tp:Packaging tp:id="CompanyB_ResponsePackage">
4562    <tp:ProcessingCapabilities
4563      tp:parse="true"
4564      tp:generate="true"/>
4565    <tp:CompositeList>
4566      <tp:Composite
4567        tp:id="CompanyB_ResponseMsg"
```

```
4568      tp:mimetype="multipart/related"
4569      tp:mimeparameters="type=text/xml">
4570      <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4571      <tp:Constituent tp:idref="CompanyB_Response"/>
4572    </tp:Composite>
4573  </tp:CompositeList>
4574</tp:Packaging>
4575  <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
4576  <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
4577    <tp:ProcessingCapabilities
4578      tp:parse="true"
4579      tp:generate="true"/>
4580    <tp:CompositeList>
4581      <tp:Composite
4582        tp:id="CompanyB_ReceiptAcknowledgmentMsg"
4583        tp:mimetype="multipart/related"
4584        tp:mimeparameters="type=text/xml">
4585        <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4586        <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
4587      </tp:Composite>
4588    </tp:CompositeList>
4589  </tp:Packaging>
4590  <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
4591  <tp:Packaging tp:id="CompanyB_ExceptionPackage">
4592    <tp:ProcessingCapabilities
4593      tp:parse="true"
4594      tp:generate="true"/>
4595    <tp:CompositeList>
4596      <tp:Composite
4597        tp:id="CompanyB_ExceptionMsg"
4598        tp:mimetype="multipart/related"
4599        tp:mimeparameters="type=text/xml">
4600        <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4601        <tp:Constituent tp:idref="CompanyB_Exception"/>
4602      </tp:Composite>
4603    </tp:CompositeList>
4604  </tp:Packaging>
4605  <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
4606      or an ebXML message with an Exception signal -->
4607  <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
4608    <tp:ProcessingCapabilities
4609      tp:parse="true"
4610      tp:generate="true"/>
4611    <tp:CompositeList>
4612      <tp:Composite
4613        tp:id="CompanyB_SignalAndResponseMsg"
4614        tp:mimetype="multipart/related"
4615        tp:mimeparameters="type=text/xml">
4616        <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4617        <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
4618        <tp:Constituent tp:idref="CompanyB_Response"/>
4619      </tp:Composite>
4620    </tp:CompositeList>
4621    <tp:CompositeList>
4622      <tp:Composite
4623        tp:id="CompanyB_SyncExceptionMsg"
4624        tp:mimetype="multipart/related"
4625        tp:mimeparameters="type=text/xml">
4626        <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4627        <tp:Constituent tp:idref="CompanyB_Exception"/>
4628      </tp:Composite>
4629    </tp:CompositeList>
4630  </tp:Packaging>
4631  <tp:Comment xml:lang="en-US">Seller's Collaboration Protocol Profile</tp:Comment>
4632</tp:CollaborationProtocolProfile>
4633
```

## 4634 Appendix B Example of CPA Document (Non-Normative)

4635 The example in this appendix is to be parsed with an XML Schema parser. The schema is  
4636 available as an ASCII file at

4637 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpp-cpa-017.xsd>

4638  
4639 The example that can be parsed with the XSD is available at

4640 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpa-example-017.xml>

```
4641 <?xml version="1.0"?>
4642 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
4643 <tp:CollaborationProtocolAgreement
4644   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
4645   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4646   xmlns:xlink="http://www.w3.org/1999/xlink"
4647   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4648   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4649   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
4650           draft-cpp-cpa-017.xsd"
4651   tp:cpaid="uri:companyA-and-companyB-cpa" tp:version="017">
4652     <tp:Status tp:value="proposed"/>
4653     <tp:Start>2001-05-20T07:21:00Z</tp:Start>
4654     <tp:End>2002-05-20T07:21:00Z</tp:End>
4655     <tp:ConversationConstraints tp:invocationLimit="100" tp:concurrentConversations="10"/>
4656     <!-- Party info for CompanyA -->
4657     <tp:PartyInfo
4658       tp:partyName="CompanyA"
4659       tp:defaultMshChannelId="asyncChannelA1"
4660       tp:defaultMshPackageId="CompanyA_MshSignalPackage">
4661       <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">123456789</tp:PartyId>
4662       <tp:PartyRef xlink:href="http://CompanyA.com/about.html"/>
4663       <tp:CollaborationRole>
4664         <tp:ProcessSpecification
4665           tp:version="2.0"
4666           tp:name="PIP3A4RequestPurchaseOrder"
4667           xlink:type="simple"
4668           xlink:href="http://www.rosettanet.org/processes/3A4.xml"
4669           tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
4670         <tp:Role
4671           tp:name="Buyer"
4672           xlink:type="simple"
4673           xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
4674         <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
4675         <tp:ServiceBinding>
4676           <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
4677           <tp:CanSend>
4678             <tp:ThisPartyActionBinding
4679               tp:id="companyA_ABID1"
4680               tp:action="Purchase Order Request Action"
4681               tp:packageId="CompanyA_RequestPackage">
4682                 <tp:BusinessTransactionCharacteristics
4683                   tp:isNonRepudiationRequired="true"
4684                   tp:isNonRepudiationReceiptRequired="true"
4685                   tp:isSecureTransportRequired="true"
4686                   tp:isConfidential="transient"
4687                   tp:isAuthenticated="persistent"
4688                   tp:isTamperProof="persistent"
4689                   tp:isAuthorizationRequired="true"
4690                   tp:timeToAcknowledgeReceipt="PT2H"
4691                   tp:timeToPerform="P1D"/>
4692                 <tp:ActionContext
4693                   tp:binaryCollaboration="Request Purchase Order"
4694                   tp:businessTransactionActivity="Request Purchase Order"
4695                   tp:requestOrResponseAction="Purchase Order Request Action"/>
4696                 <tp:ChannelId>asyncChannelA1</tp:ChannelId>
```

```
4698      </tp:ThisPartyActionBinding>
4699      <tp:OtherPartyActionBinding>companyB_ABID4</tp:OtherPartyActionBinding>
4700    </tp:CanSend>
4701    <tp:CanSend>
4702      <tp:ThisPartyActionBinding
4703        tp:id="companyA_ABID2"
4704        tp:action="ReceiptAcknowledgement"
4705        tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
4706        <tp:BusinessTransactionCharacteristics
4707          tp:isNonRepudiationRequired="true"
4708          tp:isNonRepudiationReceiptRequired="true"
4709          tp:isSecureTransportRequired="true"
4710          tp:isConfidential="transient"
4711          tp:isAuthenticated="persistent"
4712          tp:isTamperProof="persistent"
4713          tp:isAuthorizationRequired="true"
4714          tp:timeToAcknowledgeReceipt="PT2H"
4715          tp:timeToPerform="P1D"/>
4716        <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4717      </tp:ThisPartyActionBinding>
4718      <tp:OtherPartyActionBinding>companyB_ABID5</tp:OtherPartyActionBinding>
4719    </tp:CanSend>
4720    <!-- The next binding uses a synchronous delivery channel -->
4721    <tp:CanSend>
4722      <tp:ThisPartyActionBinding
4723        tp:id="companyA_ABID6"
4724        tp:action="Purchase Order Request Action"
4725        tp:packageId="CompanyA_RequestPackage">
4726        <tp:BusinessTransactionCharacteristics
4727          tp:isNonRepudiationRequired="true"
4728          tp:isNonRepudiationReceiptRequired="true"
4729          tp:isSecureTransportRequired="true"
4730          tp:isConfidential="transient"
4731          tp:isAuthenticated="persistent"
4732          tp:isTamperProof="persistent"
4733          tp:isAuthorizationRequired="true"
4734          tp:timeToAcknowledgeReceipt="PT5M"
4735          tp:timeToPerform="PT5M"/>
4736        <tp:ActionContext
4737          tp:binaryCollaboration="Request Purchase Order"
4738          tp:businessTransactionActivity="Request Purchase Order"
4739          tp:requestOrResponseAction="Purchase Order Request Action"/>
4740        <tp:ChannelId>syncChannelA1</tp:ChannelId>
4741      </tp:ThisPartyActionBinding>
4742      <tp:OtherPartyActionBinding>companyB_ABID6</tp:OtherPartyActionBinding>
4743      <tp:CanReceive>
4744        <tp:ThisPartyActionBinding
4745          tp:id="companyA_ABID7"
4746          tp:action="Purchase Order Confirmation Action"
4747          tp:packageId="CompanyA_SyncReplyPackage">
4748          <tp:BusinessTransactionCharacteristics
4749            tp:isNonRepudiationRequired="true"
4750            tp:isNonRepudiationReceiptRequired="true"
4751            tp:isSecureTransportRequired="true"
4752            tp:isConfidential="transient"
4753            tp:isAuthenticated="persistent"
4754            tp:isTamperProof="persistent"
4755            tp:isAuthorizationRequired="true"
4756            tp:timeToAcknowledgeReceipt="PT5M"
4757            tp:timeToPerform="PT5M"/>
4758          <tp:ActionContext
4759            tp:binaryCollaboration="Request Purchase Order"
4760            tp:businessTransactionActivity="Request Purchase Order"
4761            tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4762          <tp:ChannelId>syncChannelA1</tp:ChannelId>
4763        </tp:ThisPartyActionBinding>
4764        <tp:OtherPartyActionBinding>companyB_ABID7</tp:OtherPartyActionBinding>
4765        <tp:CanReceive>
4766          <tp:ThisPartyActionBinding
4767            tp:id="companyA_ABID8"
```

```
4769      tp:action="Exception"
4770      tp:packageId="CompanyA_ExceptionPackage">
4771      <tp:BusinessTransactionCharacteristics
4772          tp:isNonRepudiationRequired="true"
4773          tp:isNonRepudiationReceiptRequired="true"
4774          tp:isSecureTransportRequired="true"
4775          tp:isConfidential="transient"
4776          tp:isAuthenticated="persistent"
4777          tp:isTamperProof="persistent"
4778          tp:isAuthorizationRequired="true"
4779          tp:timeToAcknowledgeReceipt="PT5M"
4780          tp:timeToPerform="PT5M"/>
4781      <tp:ChannelId>syncChannelA1</tp:ChannelId>
4782  </tp:ThisPartyActionBinding>
4783  <tp:OtherPartyActionBinding>companyB_ABID8</tp:OtherPartyActionBinding>
4784      </tp:CanReceive>
4785  </tp:CanSend>
4786  <tp:CanReceive>
4787      <tp:ThisPartyActionBinding
4788          tp:id="companyA_ABID3"
4789          tp:action="Purchase Order Confirmation Action"
4790          tp:packageId="CompanyA_ResponsePackage">
4791          <tp:BusinessTransactionCharacteristics
4792              tp:isNonRepudiationRequired="true"
4793              tp:isNonRepudiationReceiptRequired="true"
4794              tp:isSecureTransportRequired="true"
4795              tp:isConfidential="transient"
4796              tp:isAuthenticated="persistent"
4797              tp:isTamperProof="persistent"
4798              tp:isAuthorizationRequired="true"
4799              tp:timeToAcknowledgeReceipt="PT2H"
4800              tp:timeToPerform="P1D"/>
4801  <tp:ActionContext
4802      tp:binaryCollaboration="Request Purchase Order"
4803      tp:businessTransactionActivity="Request Purchase Order"
4804      tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4805      <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4806  </tp:ThisPartyActionBinding>
4807  <tp:OtherPartyActionBinding>companyB_ABID1</tp:OtherPartyActionBinding>
4808  </tp:CanReceive>
4809  <tp:CanReceive>
4810      <tp:ThisPartyActionBinding
4811          tp:id="companyA_ABID4"
4812          tp:action="ReceiptAcknowledgment"
4813          tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
4814          <tp:BusinessTransactionCharacteristics
4815              tp:isNonRepudiationRequired="true"
4816              tp:isNonRepudiationReceiptRequired="true"
4817              tp:isSecureTransportRequired="true"
4818              tp:isConfidential="transient"
4819              tp:isAuthenticated="persistent"
4820              tp:isTamperProof="persistent"
4821              tp:isAuthorizationRequired="true"
4822              tp:timeToAcknowledgeReceipt="PT2H" tp:timeToPerform="P1D"/>
4823      <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4824  </tp:ThisPartyActionBinding>
4825  <tp:OtherPartyActionBinding>companyB_ABID2</tp:OtherPartyActionBinding>
4826  </tp:CanReceive>
4827  <tp:CanReceive>
4828      <tp:ThisPartyActionBinding
4829          tp:id="companyA_ABID5"
4830          tp:action="Exception"
4831          tp:packageId="CompanyA_ExceptionPackage">
4832          <tp:BusinessTransactionCharacteristics
4833              tp:isNonRepudiationRequired="true"
4834              tp:isNonRepudiationReceiptRequired="true"
4835              tp:isSecureTransportRequired="true"
4836              tp:isConfidential="transient"
4837              tp:isAuthenticated="persistent"
4838              tp:isTamperProof="persistent"
4839              tp:isAuthorizationRequired="true"
```

```
4840           tp:timeToAcknowledgeReceipt="PT2H"
4841           tp:timeToPerform="P1D"/>
4842           <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4843           </tp:ThisPartyActionBinding>
4844           <tp:OtherPartyActionBinding>companyB_ABID3</tp:OtherPartyActionBinding>
4845           </tp:CanReceive>
4846           </tp:ServiceBinding>
4847           </tp:CollaborationRole>
4848           <!-- Certificates used by the "Buyer" company -->
4849           <tp:Certificate tp:certId="CompanyA_AppCert">
4850             <ds:KeyInfo>
4851               <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
4852             </ds:KeyInfo>
4853           </tp:Certificate>
4854           <tp:Certificate tp:certId="CompanyA_SigningCert">
4855             <ds:KeyInfo>
4856               <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
4857             </ds:KeyInfo>
4858           </tp:Certificate>
4859           <tp:Certificate tp:certId="CompanyA_EncryptionCert">
4860             <ds:KeyInfo>
4861               <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
4862             </ds:KeyInfo>
4863           </tp:Certificate>
4864           <tp:Certificate tp:certId="CompanyA_ServerCert">
4865             <ds:KeyInfo>
4866               <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
4867             </ds:KeyInfo>
4868           </tp:Certificate>
4869           <tp:Certificate tp:certId="CompanyA_ClientCert">
4870             <ds:KeyInfo>
4871               <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
4872             </ds:KeyInfo>
4873           </tp:Certificate>
4874           <tp:Certificate tp:certId="TrustedRootCertA1">
4875             <ds:KeyInfo>
4876               <ds:KeyName>TrustedRootCertA1_Key </ds:KeyName>
4877             </ds:KeyInfo>
4878           </tp:Certificate>
4879           <tp:Certificate tp:certId="TrustedRootCertA2">
4880             <ds:KeyInfo>
4881               <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
4882             </ds:KeyInfo>
4883           </tp:Certificate>
4884           <tp:Certificate tp:certId="TrustedRootCertA3">
4885             <ds:KeyInfo>
4886               <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
4887             </ds:KeyInfo>
4888           </tp:Certificate>
4889           <tp:Certificate tp:certId="TrustedRootCertA4">
4890             <ds:KeyInfo>
4891               <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
4892             </ds:KeyInfo>
4893           </tp:Certificate>
4894           <tp:Certificate tp:certId="TrustedRootCertA5">
4895             <ds:KeyInfo>
4896               <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
4897             </ds:KeyInfo>
4898           </tp:Certificate>
4899           <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
4900             <tp:TrustAnchors>
4901               <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1"/>
4902               <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2"/>
4903               <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4"/>
4904             </tp:TrustAnchors>
4905           </tp:SecurityDetails>
4906           <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
4907             <tp:TrustAnchors>
4908               <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
4909               <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
4910             </tp:TrustAnchors>
```

```
4911    </tp:SecurityDetails>
4912    <tp:DeliveryChannel
4913        tp:channelId="asyncChannelA1"
4914        tp:transportId="transportA1"
4915        tp:docExchangeId="docExchangeA1">
4916        <tp:MessagingCharacteristics
4917            tp:syncReplyMode="none"
4918            tp:ackRequested="always"
4919            tp:ackSignatureRequested="always"
4920            tp:duplicateElimination="always"/>
4921    </tp:DeliveryChannel>
4922    <tp:DeliveryChannel
4923        tp:channelId="syncChannelA1"
4924        tp:transportId="transportA2"
4925        tp:docExchangeId="docExchangeA1">
4926        <tp:MessagingCharacteristics
4927            tp:syncReplyMode="signalsAndResponse"
4928            tp:ackRequested="always"
4929            tp:ackSignatureRequested="always"
4930            tp:duplicateElimination="always"/>
4931    </tp:DeliveryChannel>
4932    <tp:Transport tp:transportId="transportA1">
4933        <tp:TransportSender
4934            <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4935            <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4936            <tp:TransportClientSecurity
4937                <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4938                <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
4939                <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4940            </tp:TransportClientSecurity>
4941        </tp:TransportSender>
4942        <tp:TransportReceiver
4943            <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4944            <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4945            <tp:Endpoint
4946                tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/async"
4947                tp:type="allPurpose"/>
4948            <tp:TransportServerSecurity
4949                <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4950                <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
4951                <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4952            </tp:TransportServerSecurity>
4953        </tp:TransportReceiver>
4954    </tp:Transport>
4955    <tp:Transport tp:transportId="transportA2">
4956        <tp:TransportSender
4957            <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4958            <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4959            <tp:TransportClientSecurity
4960                <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4961                <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
4962                <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4963            </tp:TransportClientSecurity>
4964        </tp:TransportSender>
4965        <tp:TransportReceiver
4966            <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4967            <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4968            <tp:Endpoint
4969                tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
4970                tp:type="allPurpose"/>
4971            <tp:TransportServerSecurity
4972                <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4973                <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
4974                <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4975            </tp:TransportServerSecurity>
4976        </tp:TransportReceiver>
4977    </tp:Transport>
4978    <tp:DocExchange tp:docExchangeId="docExchangeA1">
4979        <tp:ebXMLSenderBinding tp:version="2.0">
4980            <tp:ReliableMessaging>
4981            <tp:Retries>3</tp:Retries>
```

```
4982      <tp:RetryInterval>PT2H</tp:RetryInterval>
4983      <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4984    </tp:ReliableMessaging>
4985    <tp:PersistDuration>P1D</tp:PersistDuration>
4986    <tp:SenderNonRepudiation>
4987
4988  <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4989    <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4990    <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4991    sha1</tp:SignatureAlgorithm>
4992      <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
4993    </tp:SenderNonRepudiation>
4994    <tp:SenderDigitalEnvelope>
4995      <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4996      <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4997      <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
4998    </tp:SenderDigitalEnvelope>
4999  </tp:ebXMLSenderBinding>
5000  <tp:ebXMLReceiverBinding tp:version="2.0">
5001    <tp:ReliableMessaging>
5002      <tp:Retries>3</tp:Retries>
5003      <tp:RetryInterval>PT2H</tp:RetryInterval>
5004      <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
5005    </tp:ReliableMessaging>
5006    <tp:PersistDuration>P1D</tp:PersistDuration>
5007    <tp:ReceiverNonRepudiation>
5008
5009  <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
5010    <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
5011    <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
5012    sha1</tp:SignatureAlgorithm>
5013      <tp:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
5014    </tp:ReceiverNonRepudiation>
5015    <tp:ReceiverDigitalEnvelope>
5016      <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5017      <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5018      <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
5019    </tp:ReceiverDigitalEnvelope>
5020  </tp:ebXMLReceiverBinding>
5021  </tp:DocExchange>
5022 </tp:PartyInfo>
5023 <!-- Party info for CompanyB -->
5024 <tp:PartyInfo
5025   tp:partyName="CompanyB"
5026   tp:defaultMshChannelId="asyncChannelB1"
5027   tp:defaultMshPackageId="CompanyB_MshSignalPackage">
5028   <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
5029   <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html"/>
5030   <tp:CollaborationRole>
5031     <tp:ProcessSpecification
5032       tp:version="2.0"
5033       tp:name="PIP3A4RequestPurchaseOrder"
5034       xlink:type="simple"
5035       xlink:href="http://www.rosettanet.org/processes/3A4.xml"
5036       tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
5037   <tp:Role
5038     tp:name="Seller"
5039     xlink:type="simple"
5040     xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
5041   <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert"/>
5042   <tp:ServiceBinding>
5043     <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
5044   <tp:CanSend>
5045     <tp:ThisPartyActionBinding
5046       tp:id="companyB_ABID1"
5047       tp:action="Purchase Order Confirmation Action"
5048       tp:packageId="CompanyB_ResponsePackage">
5049       <tp:BusinessTransactionCharacteristics
5050         tp:isNonRepudiationRequired="true"
5051         tp:isNonRepudiationReceiptRequired="true"
5052         tp:isSecureTransportRequired="true"
```

```
5053         tp:isConfidential="transient"
5054         tp:isAuthenticated="persistent"
5055         tp:isTamperProof="persistent"
5056         tp:isAuthorizationRequired="true"
5057         tp:timeToAcknowledgeReceipt="PT2H"
5058         tp:timeToPerform="P1D"/>
5059     <tp:ActionContext
5060         tp:binaryCollaboration="Request Purchase Order"
5061         tp:businessTransactionActivity="Request Purchase Order"
5062         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
5063     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5064     </tp:ThisPartyActionBinding>
5065     <tp:OtherPartyActionBinding>companyA_ABID3</tp:OtherPartyActionBinding>
5066   </tp:CanSend>
5067   <tp:CanSend>
5068     <tp:ThisPartyActionBinding
5069       tp:id="companyB_ABID2"
5070       tp:action="ReceiptAcknowledgement"
5071       tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
5072     <tp:BusinessTransactionCharacteristics
5073       tp:isNonRepudiationRequired="true"
5074       tp:isNonRepudiationReceiptRequired="true"
5075       tp:isSecureTransportRequired="true"
5076       tp:isConfidential="transient"
5077       tp:isAuthenticated="persistent"
5078       tp:isTamperProof="persistent"
5079       tp:isAuthorizationRequired="true"
5080       tp:timeToAcknowledgeReceipt="PT2H"
5081       tp:timeToPerform="P1D"/>
5082     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5083   </tp:ThisPartyActionBinding>
5084   <tp:OtherPartyActionBinding>companyA_ABID4</tp:OtherPartyActionBinding>
5085 </tp:CanSend>
5086 <tp:CanSend>
5087   <tp:ThisPartyActionBinding
5088     tp:id="companyB_ABID3"
5089     tp:action="Exception"
5090     tp:packageId="CompanyB_ExceptionPackage">
5091   <tp:BusinessTransactionCharacteristics
5092     tp:isNonRepudiationRequired="true"
5093     tp:isNonRepudiationReceiptRequired="true"
5094     tp:isSecureTransportRequired="true"
5095     tp:isConfidential="transient"
5096     tp:isAuthenticated="persistent"
5097     tp:isTamperProof="persistent"
5098     tp:isAuthorizationRequired="true"
5099     tp:timeToAcknowledgeReceipt="PT2H"
5100     tp:timeToPerform="P1D"/>
5101   <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5102 </tp:ThisPartyActionBinding>
5103   <tp:OtherPartyActionBinding>companyA_ABID5</tp:OtherPartyActionBinding>
5104 </tp:CanSend>
5105 <tp:CanReceive>
5106   <tp:ThisPartyActionBinding
5107     tp:id="companyB_ABID4"
5108     tp:action="Purchase Order Request Action"
5109     tp:packageId="CompanyB_RequestPackage">
5110   <tp:BusinessTransactionCharacteristics
5111     tp:isNonRepudiationRequired="true"
5112     tp:isNonRepudiationReceiptRequired="true"
5113     tp:isSecureTransportRequired="true"
5114     tp:isConfidential="transient"
5115     tp:isAuthenticated="persistent"
5116     tp:isTamperProof="persistent"
5117     tp:isAuthorizationRequired="true"
5118     tp:timeToAcknowledgeReceipt="PT2H"
5119     tp:timeToPerform="P1D"/>
5120   <tp:ActionContext
5121     tp:binaryCollaboration="Request Purchase Order"
5122     tp:businessTransactionActivity="Request Purchase Order"
5123     tp:requestOrResponseAction="Purchase Order Request Action"/>
```

```
5124      <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5125      </tp:ThisPartyActionBinding>
5126      <tp:OtherPartyActionBinding>companyA_ABID1</tp:OtherPartyActionBinding>
5127    </tp:CanReceive>
5128    <tp:CanReceive>
5129      <tp:ThisPartyActionBinding
5130        tp:id="companyB_ABID5"
5131        tp:action="ReceiptAcknowledgment"
5132        tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
5133        <tp:BusinessTransactionCharacteristics
5134          tp:isNonRepudiationRequired="true"
5135          tp:isNonRepudiationReceiptRequired="true"
5136          tp:isSecureTransportRequired="true"
5137          tp:isConfidential="transient"
5138          tp:isAuthenticated="persistent"
5139          tp:isTamperProof="persistent"
5140          tp:isAuthorizationRequired="true"
5141          tp:timeToAcknowledgeReceipt="PT2H"
5142          tp:timeToPerform="PT1D"/>
5143      <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5144      </tp:ThisPartyActionBinding>
5145      <tp:OtherPartyActionBinding>companyA_ABID2</tp:OtherPartyActionBinding>
5146    </tp:CanReceive>
5147    <!-- The next binding uses a synchronous delivery channel -->
5148    <tp:CanReceive>
5149      <tp:ThisPartyActionBinding
5150        tp:id="companyB_ABID6"
5151        tp:action="Purchase Order Request Action"
5152        tp:packageId="CompanyB_RequestPackage">
5153        <tp:BusinessTransactionCharacteristics
5154          tp:isNonRepudiationRequired="true"
5155          tp:isNonRepudiationReceiptRequired="true"
5156          tp:isSecureTransportRequired="true"
5157          tp:isConfidential="transient"
5158          tp:isAuthenticated="persistent"
5159          tp:isTamperProof="persistent"
5160          tp:isAuthorizationRequired="true"
5161          tp:timeToAcknowledgeReceipt="PT5M" tp:timeToPerform="PT5M"/>
5162      <tp:ActionContext
5163        tp:binaryCollaboration="Request Purchase Order"
5164        tp:businessTransactionActivity="Request Purchase Order"
5165        tp:requestOrResponseAction="Purchase Order Request Action"/>
5166      <tp:ChannelId>syncChannelB1</tp:ChannelId>
5167    </tp:ThisPartyActionBinding>
5168    <tp:OtherPartyActionBinding>companyA_ABID6</tp:OtherPartyActionBinding>
5169    <tp:CanSend>
5170      <tp:ThisPartyActionBinding
5171        tp:id="companyB_ABID7"
5172        tp:action="Purchase Order Confirmation Action"
5173        tp:packageId="CompanyB_SyncReplyPackage">
5174        <tp:BusinessTransactionCharacteristics
5175          tp:isNonRepudiationRequired="true"
5176          tp:isNonRepudiationReceiptRequired="true"
5177          tp:isSecureTransportRequired="true"
5178          tp:isConfidential="transient"
5179          tp:isAuthenticated="persistent"
5180          tp:isTamperProof="persistent"
5181          tp:isAuthorizationRequired="true"
5182          tp:timeToAcknowledgeReceipt="PT5M"
5183          tp:timeToPerform="PT5M"/>
5184      <tp:ActionContext
5185        tp:binaryCollaboration="Request Purchase Order"
5186        tp:businessTransactionActivity="Request Purchase Order"
5187        tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
5188      <tp:ChannelId>syncChannelB1</tp:ChannelId>
5189    </tp:ThisPartyActionBinding>
5190    <tp:OtherPartyActionBinding>companyA_ABID7</tp:OtherPartyActionBinding>
5191    <tp:CanSend>
5192    <tp:CanSend>
5193      <tp:ThisPartyActionBinding
5194        tp:id="companyB_ABID8"
```

```
5195      tp:action="Exception"
5196      tp:packageId="CompanyB_ExceptionPackage">
5197      <tp:BusinessTransactionCharacteristics
5198          tp:isNonRepudiationRequired="true"
5199          tp:isNonRepudiationReceiptRequired="true"
5200          tp:isSecureTransportRequired="true"
5201          tp:isConfidential="transient"
5202          tp:isAuthenticated="persistent"
5203          tp:isTamperProof="persistent"
5204          tp:isAuthorizationRequired="true"
5205          tp:timeToAcknowledgeReceipt="PT5M"
5206          tp:timeToPerform="PT5M"/>
5207          <tp:ChannelId>syncChannelB1</tp:ChannelId>
5208      </tp:ThisPartyActionBinding>
5209      <tp:OtherPartyActionBinding>companyA_ABID8</tp:OtherPartyActionBinding>
5210      </tp:CanSend>
5211      </tp:CanReceive>
5212      </tp:ServiceBinding>
5213  </tp:CollaborationRole>
5214  <!-- Certificates used by the "Seller" company -->
5215  <tp:Certificate tp:certId="CompanyB_AppCert">
5216      <ds:KeyInfo>
5217          <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
5218      </ds:KeyInfo>
5219  </tp:Certificate>
5220  <tp:Certificate tp:certId="CompanyB_SigningCert">
5221      <ds:KeyInfo>
5222          <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
5223      </ds:KeyInfo>
5224  </tp:Certificate>
5225  <tp:Certificate tp:certId="CompanyB_EncryptionCert">
5226      <ds:KeyInfo>
5227          <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
5228      </ds:KeyInfo>
5229  </tp:Certificate>
5230  <tp:Certificate tp:certId="CompanyB_ServerCert">
5231      <ds:KeyInfo>
5232          <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
5233      </ds:KeyInfo>
5234  </tp:Certificate>
5235  <tp:Certificate tp:certId="CompanyB_ClientCert">
5236      <ds:KeyInfo>
5237          <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
5238      </ds:KeyInfo>
5239  </tp:Certificate>
5240  <tp:Certificate tp:certId="TrustedRootCertB4">
5241      <ds:KeyInfo>
5242          <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
5243      </ds:KeyInfo>
5244  </tp:Certificate>
5245  <tp:Certificate tp:certId="TrustedRootCertB5">
5246      <ds:KeyInfo>
5247          <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
5248      </ds:KeyInfo>
5249  </tp:Certificate>
5250  <tp:Certificate tp:certId="TrustedRootCertB6">
5251      <ds:KeyInfo>
5252          <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
5253      </ds:KeyInfo>
5254  </tp:Certificate>
5255  <tp:Certificate tp:certId="TrustedRootCertB7">
5256      <ds:KeyInfo>
5257          <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
5258      </ds:KeyInfo>
5259  </tp:Certificate>
5260  <tp:Certificate tp:certId="TrustedRootCertB8">
5261      <ds:KeyInfo>
5262          <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
5263      </ds:KeyInfo>
5264  </tp:Certificate>
5265  <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
```

```
5266      <tp:TrustAnchors>
5267          <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5"/>
5268          <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6"/>
5269          <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4"/>
5270      </tp:TrustAnchors>
5271  </tp:SecurityDetails>
5272  <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
5273      <tp:TrustAnchors>
5274          <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8"/>
5275          <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7"/>
5276      </tp:TrustAnchors>
5277  </tp:SecurityDetails>
5278  <!-- An asynchronous delivery channel -->
5279  <tp:DeliveryChannel
5280      tp:channelId="asyncChannelB1"
5281      tp:transportId="transportB1"
5282      tp:docExchangeId="docExchangeB1">
5283      <tp:MessagingCharacteristics
5284          tp:syncReplyMode="none"
5285          tp:ackRequested="always"
5286          tp:ackSignatureRequested="always"
5287          tp:duplicateElimination="always"/>
5288  </tp:DeliveryChannel>
5289  <!-- A synchronous delivery channel -->
5290  <tp:DeliveryChannel
5291      tp:channelId="syncChannelB1"
5292      tp:transportId="transportB2"
5293      tp:docExchangeId="docExchangeB1">
5294      <tp:MessagingCharacteristics
5295          tp:syncReplyMode="signalsAndResponse"
5296          tp:ackRequested="always"
5297          tp:ackSignatureRequested="always"
5298          tp:duplicateElimination="always"/>
5299  </tp:DeliveryChannel>
5300  <tp:Transport tp:transportId="transportB1">
5301      <tp:TransportSender>
5302          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5303          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5304          <tp:TransportClientSecurity>
5305              <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5306              <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
5307              <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
5308          </tp:TransportClientSecurity>
5309      </tp:TransportSender>
5310      <tp:TransportReceiver>
5311          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5312          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5313          <tp:Endpoint
5314              tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
5315              tp:type="allPurpose"/>
5316          <tp:TransportServerSecurity>
5317              <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5318              <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
5319              <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
5320          </tp:TransportServerSecurity>
5321      </tp:TransportReceiver>
5322  </tp:Transport>
5323  <tp:Transport tp:transportId="transportB2">
5324      <tp:TransportSender>
5325          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5326          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5327          <tp:TransportClientSecurity>
5328              <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5329              <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
5330              <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
5331          </tp:TransportClientSecurity>
5332      </tp:TransportSender>
5333      <tp:TransportReceiver>
5334          <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5335          <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5336          <tp:Endpoint
```

```
5337     tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
5338     tp:type="allPurpose"/>
5339   <tp:TransportServerSecurity>
5340     <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5341     <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
5342     <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
5343   </tp:TransportServerSecurity>
5344   </tp:TransportReceiver>
5345 </tp:Transport>
5346 <tp:DocExchange tp:docExchangeId="docExchangeB1">
5347   <tp:ebXMLSenderBinding tp:version="2.0">
5348     <tp:ReliableMessaging>
5349       <tp:Retries>3</tp:Retries>
5350       <tp:RetryInterval>PT2H</tp:RetryInterval>
5351       <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
5352     </tp:ReliableMessaging>
5353     <tp:PersistDuration>P1D</tp:PersistDuration>
5354     <tp:SenderNonRepudiation>
5355
5356   <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
5357     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
5358     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
5359     sha1</tp:SignatureAlgorithm>
5360       <tp:SigningCertificateRef tp:certId="CompanyB_SigningCert"/>
5361     </tp:SenderNonRepudiation>
5362     <tp:SenderDigitalEnvelope>
5363       <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5364       <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5365       <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
5366     </tp:SenderDigitalEnvelope>
5367   </tp:ebXMLSenderBinding>
5368   <tp:ebXMLReceiverBinding tp:version="2.0">
5369     <tp:ReliableMessaging>
5370       <tp:Retries>3</tp:Retries>
5371       <tp:RetryInterval>PT2H</tp:RetryInterval>
5372       <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
5373     </tp:ReliableMessaging>
5374     <tp:PersistDuration>P1D</tp:PersistDuration>
5375     <tp:ReceiverNonRepudiation>
5376
5377   <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
5378     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
5379     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
5380     sha1</tp:SignatureAlgorithm>
5381       <tp:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
5382     </tp:ReceiverNonRepudiation>
5383     <tp:ReceiverDigitalEnvelope>
5384       <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5385       <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5386       <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert"/>
5387     </tp:ReceiverDigitalEnvelope>
5388   </tp:ebXMLReceiverBinding>
5389   </tp:DocExchange>
5390 </tp:PartyInfo>
5391 <!-- SimplePart corresponding to the SOAP Envelope -->
5392 <tp:SimplePart
5393   tp:id="CompanyA_MsgHdr"
5394   tp:mimetype="text/xml">
5395   <tp:NamespaceSupported
5396     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5397     tp:version="2.0">
5398     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5399   </tp:NamespaceSupported>
5400 </tp:SimplePart>
5401 <tp:SimplePart
5402   tp:id="CompanyB_MsgHdr"
5403   tp:mimetype="text/xml">
5404   <tp:NamespaceSupported
5405     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5406     tp:version="2.0">
5407     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
```

```
5408 </tp:NamespaceSupported>
5409 </tp:SimplePart>
5410 <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
5411 <tp:SimplePart
5412   tp:id="CompanyA_ReceiptAcknowledgment"
5413   tp:mimetype="application/xml">
5414   <tp:NamespaceSupported
5415     tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
5416     tp:version="2.0">http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
5417   </tp:NamespaceSupported>
5418 </tp:SimplePart>
5419 <tp:SimplePart
5420   tp:id="CompanyB_ReceiptAcknowledgment"
5421   tp:mimetype="application/xml">
5422   <tp:NamespaceSupported
5423     tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
5424     tp:version="2.0">
5425     http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
5426   </tp:NamespaceSupported>
5427 </tp:SimplePart>
5428 <!-- SimplePart corresponding to an Exception business signal -->
5429 <tp:SimplePart
5430   tp:id="CompanyA_Exception"
5431   tp:mimetype="application/xml">
5432   <tp:NamespaceSupported
5433     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5434     tp:version="2.0">
5435     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5436   </tp:NamespaceSupported>
5437 </tp:SimplePart>
5438 <tp:SimplePart
5439   tp:id="CompanyB_Exception"
5440   tp:mimetype="application/xml">
5441   <tp:NamespaceSupported
5442     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5443     tp:version="2.0">
5444     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5445   </tp:NamespaceSupported>
5446 </tp:SimplePart>
5447 <!-- SimplePart corresponding to a request action -->
5448 <tp:SimplePart
5449   tp:id="CompanyA_Request"
5450   tp:mimetype="application/xml">
5451   <tp:NamespaceSupported
5452     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
5453     tp:version="1.0">
5454     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
5455   </tp:NamespaceSupported>
5456 </tp:SimplePart>
5457 <tp:SimplePart
5458   tp:id="CompanyB_Request"
5459   tp:mimetype="application/xml">
5460   <tp:NamespaceSupported
5461     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
5462     tp:version="1.0">
5463     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
5464   </tp:NamespaceSupported>
5465 </tp:SimplePart>
5466 <!-- SimplePart corresponding to a response action -->
5467 <tp:SimplePart
5468   tp:id="CompanyA_Response"
5469   tp:mimetype="application/xml">
5470   <tp:NamespaceSupported
5471     tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
5472     tp:version="1.0">
5473     http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
5474   </tp:NamespaceSupported>
5475 </tp:SimplePart>
5476 <tp:SimplePart
5477   tp:id="CompanyB_Response"
5478   tp:mimetype="application/xml">
```

```
5479     <tp:NamespaceSupported  
5480         tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"  
5481         tp:version="1.0">  
5482         http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd  
5483     </tp:NamespaceSupported>  
5484 </tp:SimplePart>  
5485 <!-- An ebXML message with a SOAP Envelope only -->  
5486 <tp:Packaging  
5487     tp:id="CompanyA_MshSignalPackage">  
5488     <tp:ProcessingCapabilities  
5489         tp:parse="true"  
5490         tp:generate="true"/>  
5491     <tp:CompositeList>  
5492         <tp:Composite  
5493             tp:id="CompanyA_MshSignal"  
5494             tp:mimetype="multipart/related"  
5495             tp:mimeparameters="type=text/xml">  
5496             <tp:Constituent tp:idref="CompanyA_MsgHdr"/>  
5497         </tp:Composite>  
5498     </tp:CompositeList>  
5499 </tp:Packaging>  
5500 <tp:Packaging  
5501     tp:id="CompanyB_MshSignalPackage">  
5502     <tp:ProcessingCapabilities  
5503         tp:parse="true"  
5504         tp:generate="true"/>  
5505     <tp:CompositeList>  
5506         <tp:Composite  
5507             tp:id="CompanyB_MshSignal"  
5508             tp:mimetype="multipart/related"  
5509             tp:mimeparameters="type=text/xml">  
5510             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>  
5511         </tp:Composite>  
5512     </tp:CompositeList>  
5513 </tp:Packaging>  
5514 <!-- An ebXML message with a SOAP Envelope plus a request action payload -->  
5515 <tp:Packaging tp:id="CompanyA_RequestPackage">  
5516     <tp:ProcessingCapabilities  
5517         tp:parse="true"  
5518         tp:generate="true"/>  
5519     <tp:CompositeList>  
5520         <tp:Composite  
5521             tp:id="CompanyA_RequestMsg"  
5522             tp:mimetype="multipart/related"  
5523             tp:mimeparameters="type=text/xml">  
5524             <tp:Constituent tp:idref="CompanyA_MsgHdr"/>  
5525             <tp:Constituent tp:idref="CompanyA_Request"/>  
5526         </tp:Composite>  
5527     </tp:CompositeList>  
5528 </tp:Packaging>  
5529 <tp:Packaging tp:id="CompanyB_RequestPackage">  
5530     <tp:ProcessingCapabilities  
5531         tp:parse="true"  
5532         tp:generate="true"/>  
5533     <tp:CompositeList>  
5534         <tp:Composite  
5535             tp:id="CompanyB_RequestMsg"  
5536             tp:mimetype="multipart/related"  
5537             tp:mimeparameters="type=text/xml">  
5538             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>  
5539             <tp:Constituent tp:idref="CompanyB_Request"/>  
5540         </tp:Composite>  
5541     </tp:CompositeList>  
5542 </tp:Packaging>  
5543 <!-- An ebXML message with a SOAP Envelope plus a response action payload -->  
5544 <tp:Packaging tp:id="CompanyA_ResponsePackage">  
5545     <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>  
5546     <tp:CompositeList>  
5547         <tp:Composite  
5548             tp:id="CompanyA_ResponseMsg"  
5549             tp:mimetype="multipart/related"
```

```
5550      tp:mimeparameters="type=text/xml">
5551          <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5552          <tp:Constituent tp:idref="CompanyA_Response"/>
5553      </tp:Composite>
5554  </tp:CompositeList>
5555 </tp:Packaging>
5556 <tp:Packaging tp:id="CompanyB_ResponsePackage">
5557     <tp:ProcessingCapabilities
5558         tp:parse="true"
5559         tp:generate="true"/>
5560     <tp:CompositeList>
5561         <tp:Composite
5562             tp:id="CompanyB_ResponseMsg"
5563             tp:mimetype="multipart/related"
5564             tp:mimeparameters="type=text/xml">
5565                 <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5566                 <tp:Constituent tp:idref="CompanyB_Response"/>
5567             </tp:Composite>
5568         </tp:CompositeList>
5569     </tp:Packaging>
5570 <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
5571 <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
5572     <tp:ProcessingCapabilities
5573         tp:parse="true"
5574         tp:generate="true"/>
5575     <tp:CompositeList>
5576         <tp:Composite
5577             tp:id="CompanyA_ReceiptAcknowledgmentMsg"
5578             tp:mimetype="multipart/related"
5579             tp:mimeparameters="type=text/xml">
5580                 <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5581                 <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
5582             </tp:Composite>
5583         </tp:CompositeList>
5584     </tp:Packaging>
5585 <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
5586     <tp:ProcessingCapabilities
5587         tp:parse="true"
5588         tp:generate="true"/>
5589     <tp:CompositeList>
5590         <tp:Composite
5591             tp:id="CompanyB_ReceiptAcknowledgmentMsg"
5592             tp:mimetype="multipart/related"
5593             tp:mimeparameters="type=text/xml">
5594                 <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5595                 <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
5596             </tp:Composite>
5597         </tp:CompositeList>
5598     </tp:Packaging>
5599 <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
5600 <tp:Packaging tp:id="CompanyA_ExceptionPackage">
5601     <tp:ProcessingCapabilities
5602         tp:parse="true"
5603         tp:generate="true"/>
5604     <tp:CompositeList>
5605         <tp:Composite
5606             tp:id="CompanyA_ExceptionMsg"
5607             tp:mimetype="multipart/related"
5608             tp:mimeparameters="type=text/xml">
5609                 <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5610                 <tp:Constituent tp:idref="CompanyA_Exception"/>
5611             </tp:Composite>
5612         </tp:CompositeList>
5613     </tp:Packaging>
5614 <tp:Packaging tp:id="CompanyB_ExceptionPackage">
5615     <tp:ProcessingCapabilities
5616         tp:parse="true"
5617         tp:generate="true"/>
5618     <tp:CompositeList>
5619         <tp:Composite
5620             tp:id="CompanyB_ExceptionMsg"
```

```
5621      tp:mimetype="multipart/related"
5622      tp:mimeparameters="type=text/xml">
5623          <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5624          <tp:Constituent tp:idref="CompanyB_Exception"/>
5625      </tp:Composite>
5626  </tp:CompositeList>
5627</tp:Packaging>
5628  <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
5629      or an ebXML message with an Exception signal -->
5630  <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
5631      <tp:ProcessingCapabilities
5632          tp:parse="true"
5633          tp:generate="true"/>
5634  <tp:CompositeList>
5635      <tp:Composite
5636          tp:id="CompanyA_SignalAndResponseMsg"
5637          tp:mimetype="multipart/related"
5638          tp:mimeparameters="type=text/xml">
5639              <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5640              <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
5641              <tp:Constituent tp:idref="CompanyA_Response"/>
5642          </tp:Composite>
5643      </tp:CompositeList>
5644  </tp:Packaging>
5645  <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
5646      <tp:ProcessingCapabilities
5647          tp:parse="true"
5648          tp:generate="true"/>
5649  <tp:CompositeList>
5650      <tp:Composite
5651          tp:id="CompanyB_SignalAndResponseMsg"
5652          tp:mimetype="multipart/related"
5653          tp:mimeparameters="type=text/xml">
5654              <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5655              <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
5656              <tp:Constituent tp:idref="CompanyB_Response"/>
5657          </tp:Composite>
5658      </tp:CompositeList>
5659  </tp:Packaging>
5660      <tp:Comment xml:lang="en-US">buy/sell agreement between CompanyA.com and
5661      CompanyB.com</tp:Comment>
5662  </tp:CollaborationProtocolAgreement>
5663
```

## 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/draft-bpss-example-017.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/ebBPSS1.03.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 ebBPSS1.03.xsd"
  name="PIP3A4RequestPurchaseOrder"
  uid="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="Purchase 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="Purchase 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="Purchase Order Request"
        businessDocumentIDRef="Pip3A4PurchaseOrderRequest"
        isAuthenticated="persistent"
        isConfidential="transient"
        isTamperProof="persistent"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity
      name="Purchase Order Confirmation Action"
      nameID="PurchaseOrderConfirmationAction"
      isAuthorizationRequired="true"
```

```
5727     isIntelligibleCheckRequired="true"
5728     isNonRepudiationReceiptRequired="false"
5729     isNonRepudiationRequired="true"
5730     timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
5731   <DocumentEnvelope
5732     businessDocument="Purchase Order Confirmation"
5733     businessDocumentIDRef="Pip3A4PurchaseOrderConfirmation"
5734     isAuthenticated="persistent"
5735     isConfidential="transient"
5736     isPositiveResponse="true"
5737     isTamperProof="persistent"/>
5738   </RespondingBusinessActivity>
5739 </BusinessTransaction>
5740 <BinaryCollaboration
5741   name="Request Purchase Order"
5742   nameID="RequestPurchaseOrder_BC">
5743   <InitiatingRole
5744     name="Buyer"
5745     nameID="BuyerId"/>
5746   <RespondingRole
5747     name="Seller"
5748     nameID="SellerId"/>
5749   <Start toBusinessState="Request Purchase Order"/>
5750   <BusinessTransactionActivity
5751     name="Request Purchase Order"
5752     nameID="RequestPurchaseOrder_BTA"
5753     businessTransaction="Request Purchase Order"
5754     businessTransactionIDRef="RequestPurchaseOrder_BT"
5755     fromAuthorizedRole="Buyer" fromAuthorizedRoleIDRef="BuyerId"
5756     toAuthorizedRole="Seller" toAuthorizedRoleIDRef="SellerId"
5757     isLegallyBinding="true"
5758     timeToPerform="P0Y0M0DT24H0M0S"
5759     isConcurrent="false"/>
5760   <Transition
5761     fromBusinessState="Request Purchase Order"
5762     toBusinessState="Request Purchase Order"/>
5763   <Success
5764     fromBusinessState="Request Purchase Order"
5765     conditionGuard="Success"/>
5766   <Failure
5767     fromBusinessState="Request Purchase Order"
5768     conditionGuard="BusinessFailure"/>
5769   </BinaryCollaboration>
5770 </ProcessSpecification>
5771
```

## 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/draft-cpp-cpa-017.xsd>

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 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="017">
  <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="miparameters" 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>
```

```
5837      <element ref="tns:PartyInfo" maxOccurs="unbounded"/>
5838      <element ref="tns:SimplePart" maxOccurs="unbounded"/>
5839      <element ref="tns:Packaging" maxOccurs="unbounded"/>
5840      <element ref="tns:Signature" minOccurs="0"/>
5841      <element ref="tns:Comment" minOccurs="0" maxOccurs="unbounded"/>
5842    </sequence>
5843    <attribute name="cppid" type="tns:non-empty-string" use="required"/>
5844    <attribute ref="tns:version" use="required"/>
5845  </complexType>
5846</element>
5847 <element name="ProcessSpecification">
5848   <complexType>
5849     <sequence>
5850       <element ref="ds:Reference" minOccurs="0" maxOccurs="unbounded"/>
5851     </sequence>
5852     <attribute name="name" type="tns:non-empty-string" use="required"/>
5853     <attribute ref="tns:version" use="required"/>
5854     <attributeGroup ref="tns:xlink.grp"/>
5855     <attribute name="uuid" type="anyURI"/>
5856   </complexType>
5857 </element>
5858 <element name="Service" type="tns:service.type"/>
5859 <element name="Protocol" type="tns:protocol.type"/>
5860 <element name="SendingProtocol" type="tns:protocol.type"/>
5861 <element name="ReceivingProtocol" type="tns:protocol.type"/>
5862 <element name="OverrideMshActionBinding">
5863   <complexType>
5864     <attribute name="action" type="tns:non-empty-string" use="required"/>
5865     <attribute name="channelId" type="IDREF" use="required"/>
5866   </complexType>
5867 </element>
5868 <element name="ChannelId" type="IDREF"/>
5869 <complexType name="ActionBinding.type">
5870   <sequence>
5871     <element ref="tns:BusinessTransactionCharacteristics"/>
5872     <element ref="tns:ActionContext" minOccurs="0"/>
5873     <element ref="tns:ChannelId" maxOccurs="unbounded"/>
5874     <any namespace="#other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
5875   </sequence>
5876   <attribute name="id" type="ID" use="required"/>
5877   <attribute name="action" type="tns:non-empty-string" use="required"/>
5878   <attribute name="packageId" type="IDREF" use="required"/>
5879   <attribute ref="xlink:href" use="optional"/>
5880   <attribute ref="xlink:type" fixed="simple"/>
5881 </complexType>
5882 <element name="ActionContext">
5883   <complexType>
5884     <sequence>
5885       <element ref="tns:CollaborationActivity" minOccurs="0"/>
5886       <any namespace="#other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
5887     </sequence>
5888     <attribute name="binaryCollaboration" type="tns:non-empty-string" use="required"/>
5889     <attribute name="businessTransactionActivity" type="tns:non-empty-string" use="required"/>
5890     <attribute name="requestOrResponseAction" type="tns:non-empty-string" use="required"/>
5891   </complexType>
5892 </element>
5893 <element name="CollaborationActivity">
5894   <complexType>
5895     <sequence>
5896       <element ref="tns:CollaborationActivity" minOccurs="0"/>
5897     </sequence>
5898     <attribute name="name" type="tns:non-empty-string"/>
5899   </complexType>
5900 </element>
5901 <element name="CollaborationRole">
5902   <complexType>
5903     <sequence>
5904       <element ref="tns:ProcessSpecification"/>
5905       <element ref="tns:Role"/>
5906       <element name="ApplicationCertificateRef" type="tns:CertificateRef.type" minOccurs="0" maxOccurs="unbounded"/>
5907 
```

```
5908     <element name="ApplicationSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
5909     minOccurs="0"/>
5910     <element ref="tns:ServiceBinding"/>
5911     </sequence>
5912   </complexType>
5913 </element>
5914 <element name="PartyInfo">
5915   <complexType>
5916     <sequence>
5917       <element ref="tns:PartyId" maxOccurs="unbounded"/>
5918       <element ref="tns:PartyRef" maxOccurs="unbounded"/>
5919       <element ref="tns:CollaborationRole" maxOccurs="unbounded"/>
5920       <element ref="tns:Certificate" maxOccurs="unbounded"/>
5921       <element ref="tns:SecurityDetails" maxOccurs="unbounded"/>
5922       <element ref="tns:DeliveryChannel" maxOccurs="unbounded"/>
5923       <element ref="tns:Transport" maxOccurs="unbounded"/>
5924       <element ref="tns:DocExchange" maxOccurs="unbounded"/>
5925       <element ref="tns:OverrideMshActionBinding" minOccurs="0" maxOccurs="unbounded"/>
5926     </sequence>
5927     <attribute name="partyName" type="tns:non-empty-string" use="required"/>
5928     <attribute name="defaultMshChannelId" type="IDREF" use="required"/>
5929     <attribute name="defaultMshPackageId" type="IDREF" use="required"/>
5930   </complexType>
5931 </element>
5932 <element name="PartyId">
5933   <complexType>
5934     <simpleContent>
5935       <extension base="tns:non-empty-string">
5936         <attribute name="type" type="anyURI"/>
5937       </extension>
5938     </simpleContent>
5939   </complexType>
5940 </element>
5941 <element name="PartyRef">
5942   <complexType>
5943     <sequence>
5944     </sequence>
5945     <attributeGroup ref="tns:xlink.grp"/>
5946     <attribute name="type" type="anyURI"/>
5947     <attribute name="schemaLocation" type="anyURI"/>
5948   </complexType>
5949 </element>
5950 <element name="DeliveryChannel">
5951   <complexType>
5952     <sequence>
5953       <element ref="tns:MessagingCharacteristics"/>
5954     </sequence>
5955     <attribute name="channelId" type="ID" use="required"/>
5956     <attribute name="transportId" type="IDREF" use="required"/>
5957     <attribute name="docExchangeId" type="IDREF" use="required"/>
5958   </complexType>
5959 </element>
5960 <element name="Transport">
5961   <complexType>
5962     <sequence>
5963       <element ref="tns:TransportSender" minOccurs="0"/>
5964       <element ref="tns:TransportReceiver" minOccurs="0"/>
5965     </sequence>
5966     <attribute name="transportId" type="ID" use="required"/>
5967   </complexType>
5968 </element>
5969 <element name="AccessAuthentication" type="tns:accessAuthentication.type"/>
5970 <element name="TransportSender">
5971   <complexType>
5972     <sequence>
5973       <element name="TransportProtocol" type="tns:protocol.type"/>
5974       <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded"/>
5975       <element ref="tns:TransportClientSecurity" minOccurs="0"/>
5976     </sequence>
5977   </complexType>
5978 </element>
```

```
5979 <element name="TransportReceiver">
5980   <complexType>
5981     <sequence>
5982       <element name="TransportProtocol" type="tns:protocol.type"/>
5983       <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded"/>
5984       <element ref="tns:Endpoint" maxOccurs="unbounded"/>
5985       <element ref="tns:TransportServerSecurity" minOccurs="0"/>
5986     </sequence>
5987   </complexType>
5988 </element>
5989 <element name="Endpoint">
5990   <complexType>
5991     <attribute name="uri" type="anyURI" use="required"/>
5992     <attribute name="type" type="tns:endpointType.type" default="allPurpose"/>
5993   </complexType>
5994 </element>
5995 <element name="TransportClientSecurity">
5996   <complexType>
5997     <sequence>
5998       <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
5999       <element name="ClientCertificateRef" type="tns:CertificateRef.type" minOccurs="0"/>
6000       <element name="ServerSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6001       minOccurs="0"/>
6002         <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
6003       </sequence>
6004     </complexType>
6005   </element>
6006 <element name="TransportServerSecurity">
6007   <complexType>
6008     <sequence>
6009       <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
6010       <element name="ServerCertificateRef" type="tns:CertificateRef.type"/>
6011       <element name="ClientSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6012       minOccurs="0"/>
6013         <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
6014       </sequence>
6015     </complexType>
6016   </element>
6017 <element name="Certificate">
6018   <complexType>
6019     <sequence>
6020       <element ref="ds:KeyInfo"/>
6021     </sequence>
6022       <attribute name="certId" type="ID" use="required"/>
6023     </complexType>
6024   </element>
6025 <element name="DocExchange">
6026   <complexType>
6027     <sequence>
6028       <element ref="tns:ebXMLSenderBinding" minOccurs="0"/>
6029       <element ref="tns:ebXMLReceiverBinding" minOccurs="0"/>
6030     </sequence>
6031       <attribute name="docExchangeId" type="ID" use="required"/>
6032     </complexType>
6033   </element>
6034 <element name="ReliableMessaging">
6035   <complexType>
6036     <sequence>
6037       <element name="Retries" type="integer" minOccurs="0"/>
6038       <element name="RetryInterval" type="duration" minOccurs="0"/>
6039       <element name="MessageOrderSemantics" type="tns:messageOrderSemantics.type"/>
6040     </sequence>
6041   </complexType>
6042 </element>
6043 <element name="PersistDuration" type="duration"/>
6044 <element name="SenderNonRepudiation">
6045   <complexType>
6046     <sequence>
6047       <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
6048       <element ref="tns:HashFunction"/>
6049       <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>

```

```
6050      <element name="SigningCertificateRef" type="tns:CertificateRef.type"/>
6051      </sequence>
6052    </complexType>
6053  </element>
6054  <element name="ReceiverNonRepudiation">
6055    <complexType>
6056      <sequence>
6057        <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
6058        <element ref="tns:HashFunction"/>
6059        <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>
6060        <element name="SigningSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6061 minOccurs="0"/>
6062      </sequence>
6063    </complexType>
6064  </element>
6065  <element name="HashFunction" type="tns:non-empty-string"/>
6066  <element name="EncryptionAlgorithm">
6067    <complexType>
6068      <simpleContent>
6069        <extension base="tns:non-empty-string">
6070          <attribute name="minimumStrength" type="integer"/>
6071          <attribute name="oid" type="tns:non-empty-string"/>
6072          <attribute name="w3c" type="tns:non-empty-string"/>
6073          <attribute name="enumerationType" type="tns:non-empty-string"/>
6074        </extension>
6075      </simpleContent>
6076    </complexType>
6077  </element>
6078  <element name="SignatureAlgorithm">
6079    <complexType>
6080      <simpleContent>
6081        <extension base="tns:non-empty-string">
6082          <attribute name="oid" type="tns:non-empty-string"/>
6083          <attribute name="w3c" type="tns:non-empty-string"/>
6084          <attribute name="enumerationType" type="tns:non-empty-string"/>
6085        </extension>
6086      </simpleContent>
6087    </complexType>
6088  </element>
6089  <element name="SenderDigitalEnvelope">
6090    <complexType>
6091      <sequence>
6092        <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
6093        <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
6094        <element name="EncryptionSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6095 minOccurs="0"/>
6096      </sequence>
6097    </complexType>
6098  </element>
6099  <element name="ReceiverDigitalEnvelope">
6100    <complexType>
6101      <sequence>
6102        <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
6103        <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
6104        <element name="EncryptionCertificateRef" type="tns:CertificateRef.type"/>
6105      </sequence>
6106    </complexType>
6107  </element>
6108  <element name="ebXMLSenderBinding">
6109    <complexType>
6110      <sequence>
6111        <element ref="tns:ReliableMessaging" minOccurs="0"/>
6112        <element ref="tns:PersistDuration" minOccurs="0"/>
6113        <element ref="tns:SenderNonRepudiation" minOccurs="0"/>
6114        <element ref="tns:SenderDigitalEnvelope" minOccurs="0"/>
6115        <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
6116      </sequence>
6117      <attribute ref="tns:version" use="required"/>
6118    </complexType>
6119  </element>
6120  <element name="ebXMLReceiverBinding">
```

```
6121 <complexType>
6122   <sequence>
6123     <element ref="tns:ReliableMessaging" minOccurs="0"/>
6124     <element ref="tns:PersistDuration" minOccurs="0"/>
6125     <element ref="tns:ReceiverNonRepudiation" minOccurs="0"/>
6126     <element ref="tns:ReceiverDigitalEnvelope" minOccurs="0"/>
6127     <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
6128   </sequence>
6129   <attribute ref="tns:version" use="required"/>
6130 </complexType>
6131 </element>
6132 <element name="NamespaceSupported">
6133   <complexType>
6134     <simpleContent>
6135       <extension base="anyURI">
6136         <attribute name="location" type="anyURI" use="required"/>
6137         <attribute ref="tns:version"/>
6138       </extension>
6139     </simpleContent>
6140   </complexType>
6141 </element>
6142 <element name="BusinessTransactionCharacteristics">
6143   <complexType>
6144     <attribute name="isNonRepudiationRequired" type="boolean"/>
6145     <attribute name="isNonRepudiationReceiptRequired" type="boolean"/>
6146     <attribute name="isSecureTransportRequired" type="boolean"/>
6147     <attribute name="isConfidential" type="tns:persistenceLevel.type"/>
6148     <attribute name="isAuthenticated" type="tns:persistenceLevel.type"/>
6149     <attribute name="isTamperProof" type="tns:persistenceLevel.type"/>
6150     <attribute name="isAuthorizationRequired" type="boolean"/>
6151     <attribute name="isIntelligibleCheckRequired" type="boolean"/>
6152     <attribute name="timeToAcknowledgeReceipt" type="duration"/>
6153     <attribute name="timeToAcknowledgeAcceptance" type="duration"/>
6154     <attribute name="timeToPerform" type="duration"/>
6155     <attribute name="retryCount" type="integer"/>
6156   </complexType>
6157 </element>
6158 <element name="MessagingCharacteristics">
6159   <complexType>
6160     <attribute ref="tns:syncReplyMode" default="none"/>
6161     <attribute name="ackRequested" type="tns:perMessageCharacteristics.type"
6162     default="perMessage"/>
6163     <attribute name="ackSignatureRequested" type="tns:perMessageCharacteristics.type"
6164     default="perMessage"/>
6165     <attribute name="duplicateElimination" type="tns:perMessageCharacteristics.type"
6166     default="perMessage"/>
6167     <attribute name="actor" type="tns:actor.type"/>
6168   </complexType>
6169 </element>
6170 <element name="ServiceBinding">
6171   <complexType>
6172     <sequence>
6173       <element ref="tns:Service"/>
6174       <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded"/>
6175       <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded"/>
6176     </sequence>
6177   </complexType>
6178 </element>
6179 <element name="CanSend">
6180   <complexType>
6181     <sequence>
6182       <element name="ThisPartyActionBinding" type="tns:ActionBinding.type"/>
6183       <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0"/>
6184       <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded"/>
6185     </sequence>
6186   </complexType>
6187 </element>
6188 <element name="CanReceive">
6189   <complexType>
6190     <sequence>
6191       <element name="ThisPartyActionBinding" type="tns:ActionBinding.type"/>
```

```
6192      <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0"/>
6193      <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded"/>
6194    </sequence>
6195  </complexType>
6196</element>
6197 <element name="Status">
6198   <complexType>
6199     <attribute name="value" type="tns:statusValue.type" use="required"/>
6200   </complexType>
6201</element>
6202 <element name="Start" type="dateTime"/>
6203 <element name="End" type="dateTime"/>
6204 <element name="Type" type="tns:non-empty-string"/>
6205 <element name="ConversationConstraints">
6206   <complexType>
6207     <attribute name="invocationLimit" type="int"/>
6208     <attribute name="concurrentConversations" type="int"/>
6209   </complexType>
6210</element>
6211 <element name="Role">
6212   <complexType>
6213     <attribute name="name" type="tns:non-empty-string" use="required"/>
6214     <attributeGroup ref="tns:xlink.grp"/>
6215   </complexType>
6216</element>
6217 <element name="SignatureTransforms">
6218   <complexType>
6219     <sequence>
6220       <element ref="ds:Transform" maxOccurs="unbounded"/>
6221     </sequence>
6222   </complexType>
6223</element>
6224 <element name="EncryptionTransforms">
6225   <complexType>
6226     <sequence>
6227       <element ref="ds:Transform" maxOccurs="unbounded"/>
6228     </sequence>
6229   </complexType>
6230</element>
6231 <element name="Constituent">
6232   <complexType>
6233     <sequence>
6234       <element ref="tns:SignatureTransforms" minOccurs="0"/>
6235       <element ref="tns:EncryptionTransforms" minOccurs="0"/>
6236     </sequence>
6237     <attribute ref="tns:idref" use="required"/>
6238     <attribute name="excludedFromSignature" type="boolean" default="false"/>
6239     <attribute name="minOccurs" type="nonNegativeInteger"/>
6240     <attribute name="maxOccurs" type="nonNegativeInteger"/>
6241   </complexType>
6242</element>
6243 <element name="Packaging">
6244   <complexType>
6245     <sequence>
6246       <element name="ProcessingCapabilities">
6247         <complexType>
6248           <attribute name="parse" type="boolean" use="required"/>
6249           <attribute name="generate" type="boolean" use="required"/>
6250         </complexType>
6251       </element>
6252       <element name="CompositeList" maxOccurs="unbounded">
6253         <complexType>
6254           <choice maxOccurs="unbounded">
6255             <element name="Encapsulation">
6256               <complexType>
6257                 <sequence>
6258                   <element ref="tns:Constituent"/>
6259                 </sequence>
6260               <attributeGroup ref="tns:pkg.grp"/>
6261             </complexType>
6262           </element>
6263         </choice>
6264       </element>
6265     </sequence>
6266   </complexType>
6267 </element>
```

```
6263     <element name="Composite">
6264         <complexType>
6265             <sequence>
6266                 <element ref="tns:Constituent" maxOccurs="unbounded"/>
6267             </sequence>
6268             <attributeGroup ref="tns:pkg.grp"/>
6269         </complexType>
6270     </element>
6271     </choice>
6272 </complexType>
6273 </element>
6274 </sequence>
6275     <attribute ref="tns:id" use="required"/>
6276 </complexType>
6277 </element>
6278 <element name="Comment">
6279     <complexType>
6280         <simpleContent>
6281             <extension base="tns:non-empty-string">
6282                 <attribute ref="xml:lang"/>
6283             </extension>
6284         </simpleContent>
6285     </complexType>
6286 </element>
6287 <element name="SimplePart">
6288     <complexType>
6289         <sequence>
6290             <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
6291         </sequence>
6292         <attributeGroup ref="tns:pkg.grp"/>
6293         <attribute ref="xlink:role"/>
6294     </complexType>
6295 </element>
6296 <!-- COMMON -->
6297 <simpleType name="statusValue.type">
6298     <restriction base="NMTOKEN">
6299         <enumeration value="agreed"/>
6300         <enumeration value="signed"/>
6301         <enumeration value="proposed"/>
6302     </restriction>
6303 </simpleType>
6304 <simpleType name="endpointType.type">
6305     <restriction base="NMTOKEN">
6306         <enumeration value="login"/>
6307         <enumeration value="request"/>
6308         <enumeration value="response"/>
6309         <enumeration value="error"/>
6310         <enumeration value="allPurpose"/>
6311     </restriction>
6312 </simpleType>
6313 <simpleType name="non-empty-string">
6314     <restriction base="string">
6315         <minLength value="1"/>
6316     </restriction>
6317 </simpleType>
6318 <simpleType name="syncReplyMode.type">
6319     <restriction base="NMTOKEN">
6320         <enumeration value="mshSignalsOnly"/>
6321         <enumeration value="responseOnly"/>
6322         <enumeration value="signalsAndResponse"/>
6323         <enumeration value="signalsOnly"/>
6324         <enumeration value="none"/>
6325     </restriction>
6326 </simpleType>
6327 <complexType name="service.type">
6328     <simpleContent>
6329         <extension base="tns:non-empty-string">
6330             <attribute name="type" type="tns:non-empty-string"/>
6331         </extension>
6332     </simpleContent>
6333 </complexType>
```

```
6334 <complexType name="protocol.type">
6335   <simpleContent>
6336     <extension base="tns:non-empty-string">
6337       <attribute ref="tns:version"/>
6338     </extension>
6339   </simpleContent>
6340 </complexType>
6341 <attribute name="idref" type="IDREF"/>
6342 <attribute name="id" type="ID"/>
6343 <attribute name="version" type="tns:non-empty-string"/>
6344 <attribute name="syncReplyMode" type="tns:syncReplyMode.type"/>
6345 <complexType name="SecurityPolicy.type"/>
6346 <complexType name="CertificateRef.type">
6347   <attribute name="certId" type="IDREF" use="required"/>
6348 </complexType>
6349 <simpleType name="perMessageCharacteristics.type">
6350   <restriction base="NMTOKEN">
6351     <enumeration value="always"/>
6352     <enumeration value="never"/>
6353     <enumeration value="perMessage"/>
6354   </restriction>
6355 </simpleType>
6356 <simpleType name="actor.type">
6357   <restriction base="NMTOKEN">
6358     <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:nextMSH"/>
6359     <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"/>
6360   </restriction>
6361 </simpleType>
6362 <simpleType name="messageOrderSemantics.type">
6363   <restriction base="Name">
6364     <enumeration value="Guaranteed"/>
6365     <enumeration value="NotGuaranteed"/>
6366   </restriction>
6367 </simpleType>
6368 <complexType name="SecurityDetailsRef.type">
6369   <attribute name="securityId" type="IDREF" use="required"/>
6370 </complexType>
6371 <simpleType name="persistenceLevel.type">
6372   <restriction base="Name">
6373     <enumeration value="none"/>
6374     <enumeration value="transient"/>
6375     <enumeration value="persistent"/>
6376     <enumeration value="transient-and-persistent"/>
6377   </restriction>
6378 </simpleType>
6379 <element name="SecurityDetailsRef" type="tns:SecurityDetailsRef.type"/>
6380 <element name="SecurityDetails">
6381   <complexType>
6382     <sequence>
6383       <element ref="tns:TrustAnchors" minOccurs="0"/>
6384       <element ref="tns:SecurityPolicy" minOccurs="0"/>
6385     </sequence>
6386     <attribute name="securityId" type="ID" use="required"/>
6387   </complexType>
6388 </element>
6389 <element name="TrustAnchors">
6390   <complexType>
6391     <sequence>
6392       <element name="AnchorCertificateRef" type="tns:CertificateRef.type"
maxOccurs="unbounded"/>
6393     </sequence>
6394   </complexType>
6395 </element>
6396 <element name="SecurityPolicy">
6397   <complexType>
6398     <sequence>
6399     </sequence>
6400   </complexType>
6401 </element>
6402 <simpleType name="accessAuthentication.type">
6403   <restriction base="NMTOKEN">
```

```
6405      <enumeration value="basic"/>
6406      <enumeration value="digest"/>
6407    </restriction>
6408  </simpleType>
6409</schema>
6410
```

## 6411 **Appendix E CPA Composition (Non-Normative)**

### 6412 **E.1 Suggestions for Design of Computational Procedures**

6413 A quick inspection of the schemas for the top level elements, *CollaborationProtocolProfile*  
6414 (*CPP*) and *CollaborationProtocolAgreement* (*CPA*), shows that a *CPA* can be viewed as a  
6415 result of merging portions of the *PartyInfo* elements found in constituent *CPPs*, and then  
6416 integrating these *PartyInfo* elements with other *CPA* sibling elements, such as those governing  
6417 the *CPA* validity period.

6418 Merging *CPPs* into *CPAs* is one way in which trading partners can arrive at a proposed or  
6419 “draft” *CPA*. A draft *CPA* might also be formed from a *CPA* template. A *CPA*-template  
6420 represents one party’s proposed implementation of a business process that uses placeholder  
6421 values for the identifying aspects of the other party, such as *PartyId* or *TransportEndpoint*  
6422 elements. To form a *CPA* from a *CPA* template, the placeholder values are replaced by the actual  
6423 values for the other trading partner. The actual values could themselves be extracted from the  
6424 other trading partner’s *CPP*, if one is available, or they could be obtained from an administrator  
6425 performing data entry functions.

6426 We call objects draft *CPAs* to indicate their potential use as inputs to a *CPA* negotiation process  
6427 in which a draft *CPA* is verified as suitable for both parties, modified until a suitable *CPA* is  
6428 found, or discovered to not be feasible until one side (or both) acquires additional software  
6429 capabilities. These negotiation procedures and protocols are currently being designed, their  
6430 requirements having been defined, and the resulting specifications should be available with the  
6431 next release of this specification. In general, a draft *CPA* will constitute a proposal about an  
6432 overall binding of a business process to a delivery implementation, while negotiation will be  
6433 used to arrive at detailed values for parameters reflecting a final agreement. A special companion  
6434 document, the *NegotiationDescriptorDocument*, provides both focus on what parameters can be  
6435 negotiated as well as ranges or sets of acceptable values for those parameters.

6436 In the remainder of this appendix, the goal will be to identify and describe the basic tasks that  
6437 computational procedures for the assembly of the draft *CPA* would normally accomplish. While  
6438 no normative specification is provided for an algorithm for *CPA* formation, some guidance for  
6439 implementers is provided. This information might assist the software implementer in designing a  
6440 partially automated and partially interactive software system useful for configuring business  
6441 collaboration so as to arrive at satisfactorily complete levels of interoperability.

6442 Before enumerating and describing the basic tasks, it is worthwhile mentioning two basic reasons  
6443 why we focus on the component tasks involved in *CPA* formation rather than attempt to provide  
6444 an algorithm for *CPA* formation. These reasons provide some hints to implementers about ways  
6445 in which they might customize their approaches to drafting *CPAs* from *CPPs*.

#### 6451 **E.1.1 Variability in Inputs**

6452 User preferences provide one source of variability in the inputs to the *CPA* formation process.  
6453 Let us suppose in this section that each of the *Parties* has made its *CPP* available to potential  
6454 collaborators. Normally one *Party* will have a desired business collaboration (defined in a  
6455 *Collaboration-Protocol Profile and Agreement Specification*

6455   **ProcessSpecification** document) to implement with its intended collaborator. So the information  
6456   inputs will normally involve a user preference about intended business collaboration in addition  
6457   to just the *CPPs*.

6458  
6459   A *CPA* formation tool can have access to local user information not advertised in the *CPP* that  
6460   can contribute to the *CPA* that is formed. A user can have chosen to only advertise those system  
6461   capabilities that reflect capabilities that have not been deprecated. For example, a user can only  
6462   advertise HTTP and omit FTP, even when capable of using FTP. The reason for omitting FTP  
6463   might be concerns about the scalability of managing user accounts, directories, and passwords  
6464   for FTP sessions. Despite not advertising an FTP capability, configuration software can use tacit  
6465   knowledge about its own FTP capability to form a *CPA* with an intended collaborator who  
6466   happens to have only an FTP capability for implementing a desired business collaboration. In  
6467   other words, business interests can, in this case, override the deprecation policy. Both tacit  
6468   knowledge and detailed preference information account for variability in inputs into the *CPA*  
6469   formation process.

6470  
6471   **E.1.2 Variable Stringency in Evaluating Proposed Agreements**

6472   The conditions for output of a *CPA* given two *CPPs* can involve different levels and extents of  
6473   interoperability. In other words, when an optimal solution that satisfies every level of  
6474   requirement and every other additional constraint does not exist, a *Party* can propose a *CPA* that  
6475   satisfies enough of the requirements for "a good enough" implementation. User input can be  
6476   solicited to determine what is a good enough implementation, and so can be as varied as there  
6477   are user configuration options to express preferences. In practice, compromises can be made on  
6478   security, reliable messaging, levels of signals and acknowledgments, and other matters in order  
6479   to find some acceptable means of doing business.

6480  
6481   A *CPA* can support a fully interoperable configuration in which agreement has been reached on  
6482   all technical levels needed for business collaboration. In such a case, matches in capabilities will  
6483   have been found in all relevant technical levels.

6484  
6485   However, there can be interoperable configurations agreed to in a *CPA* in which not all aspects  
6486   of a business collaboration match. Gaps can exist in packaging, security, signaling, reliable  
6487   messaging and other areas and yet the systems can still transport the business data, and special  
6488   means can be employed to handle the exceptions. In such situations, a *CPA* can reflect  
6489   configured policies or expressly solicited user permission to ignore some shortcomings in  
6490   configurations. A system might not be capable of responding in a business collaboration so as to  
6491   support a specified ability to supply non-repudiation of receipt, but might still be acceptable for  
6492   business reasons. A system might not be able to handle all the processing needed to support, for  
6493   example, SOAP with Attachments and yet still be able to treat the multipart according to  
6494   "multipart/mixed" handling and allow business collaboration to take place. In fact, short of a  
6495   failure to be able to transport data and a failure to be able to provide data relevant to the business  
6496   collaboration, there are few features that might not be temporarily or indefinitely compromised  
6497   about, given overriding *business* interests. This situation of "partial interoperability" is to be  
6498   expected to persist for some time, and so interferes with formulating a "clean" algorithm for  
6499   deciding on what is sufficient for interoperability.

## 6501 E.2 CPA Formation Component Tasks

6502 Technically viewed, a *CPA* provides "bindings" between business collaboration specifications  
6503 (such as those defined within the *ProcessSpecification*'s referenced documents) and those  
6504 services and protocols that are used to implement these specifications. The implementation takes  
6505 place at several levels and involves varied services at these levels. A *CPA* that arrives at a fully  
6506 interoperable collaboration binding can be thought of as arriving at interoperable, application-to-  
6507 application integration. *CPAs* can fall short of this goal and still be both useful and acceptable to  
6508 the collaborating *parties*. Certainly, if no matching data-transport capabilities can be discovered,  
6509 a *CPA* would not provide much in the way of interoperable integration. Likewise, partial *CPAs*  
6510 can leave significant system work to be done before a completely satisfactory application-to-  
6511 application integration is realized. Even so, partial integration can be sufficient to allow  
6512 collaboration, and to enjoy payoffs from increased levels of automation.

6513  
6514 In practice, the *CPA* formation process can produce a complete *CPA*, a failure result, a gap list  
6515 that drives a dialog with the user, or perhaps even a *CPA* that implements partial interoperability  
6516 "good enough" for the business collaborators. Because both matching capabilities and  
6517 interoperability can be matters of degree, the constituent tasks are finding the matches in  
6518 capabilities at different levels and for different services. We next proceed to characterize the  
6519 most important of these constituent tasks.  
6520

## 6521 E.3 CPA Formation from CPPs: Context of Tasks

6522 To simplify discussion, assume in the following that we are viewing the tasks faced by a  
6523 software agent when:

- 6525 1. an intended collaborator is known and the collaborator's *CPP* has been retrieved,
- 6526 2. the *ProcessSpecification* between our side and our intended collaborator has been  
6527 selected,
- 6528 3. the *Service*, *Action*, and the specific *Role* elements that our software agent is to play in  
6529 the business collaboration is known, and
- 6530 4. finally, the capabilities that we have advertised in our *CPP* are known

6531 For vividness, we will develop our discussions using the "3A4" ebBPSS example and the *CPPs*  
6532 of Company A and B that are found in full in appendices of this document and that should also  
6533 be available at the web site for the OASIS ebXML CPPA Technical Committee. For simplicity,  
6534 we will assume that the information about capabilities is restricted to what is available in our  
6535 agent's *CPP*, and in the *CPP* of our intended collaborator. We will suppose that we have taken  
6536 on the viewpoint of Company A assembling a draft *CPA*. Please note that there is no guarantee  
6537 that the same draft *CPAs* will be produced in the same order from differing viewpoints.  
6538

6539 In general, the basic tasks consist of finding "matches" between our capabilities and our intended  
6540 collaborator's capabilities at the various levels of the collaboration protocol stack and with  
6541 respect to the services supplied at these various levels. This stack, which need not be  
6542 characterized in any detail, is at least distinguished by an application level and a messaging  
6543 transfer level. The application level is governed by a business process flow specification, such as  
6544 ebBPSS. The messaging transfer level will consist of a number of requirements and options  
6545

6546 concerning transfer protocols, security, packaging, and messaging patterns (such as various kinds  
6547 of acknowledgment, error messages, and the like.)

6548  
6549 In actually assembling the tasks into a computational process, it will generally make sense to  
6550 perform the tasks in a certain order. The overall order reflects the implicit structure of the *CPA*:  
6551 first undertake those tasks to ensure that there is a match with respect to the business  
6552 collaboration process. Without finding that the collaborators can participate in the same  
6553 **ProcessSpecification** successfully, there is little point in working through implementation  
6554 options. Then, examine the matches within the components of the bindings that have been  
6555 announced for the business collaboration process, checking for the most indispensable “matches”  
6556 first (**Transport**-related), and continuing checks on the other layers reflecting integrated  
6557 interoperability at packaging, security, signals and protocol patterns, and so on. With this basic  
6558 overview in mind, let us proceed to consider the basic tasks in greater detail.  
6559

#### 6560 **E.4 Business Collaboration Process Matching Tasks**

6561 Company A has announced within its *CPP*, at least one **PartyInfo** element. For current purposes,  
6562 the most important initial focus is on all the sibling elements with the path  
6563 **/CollaborationProtocolProfile/PartyInfo/CollaborationRole**. Each element of this kind has a  
6564 child, **ProcessSpecification**. Our initial matching task (probably better viewed as a filtering  
6565 task) is to select those nodes where the **ProcessSpecification** is one that we are interested in  
6566 building a CPA for! Checking the attribute values allows us to select by comparing values in the  
6567 **name**, **xlink:href** or **uuid** attributes. The definitive value for matching ebBPSS process  
6568 specifications is the value found in the **ProcessSpecification/@uuid** attribute.  
6569

##### 6570 **E.4.1 Matching ProcessSpecification/Roles, and Actions: Initial Filtering and Selection**

6571 The previous task has essentially found two **CollaborationRole** node sets within our and our  
6572 collaborator’s CPP documents where the **ProcessSpecifications** are identical, and equal to the  
6573 value of interest given above. In other words, we have **CollaborationRoles** with  
6574 **ProcessSpecification/@name='PIP3A4RequestPurchaseOrder'**. It is convenient but not essential  
6575 to use the **name** attribute in performing this selection.  
6576

6577 We next proceed to filter these node sets. We have been given our **Role** element value for our  
6578 participation in the ProcessSpecification. For Company A, this **Role** has the **name** attribute with  
6579 value ‘Buyer’. Because we are here considering only **BinaryCollaborations** in ebBPSS  
6580 terminology (or their equivalent in other flow languages), we are only interested in those  
6581 **CollaborationRole** node sets within our collaborator’s *CPP* that have a **Role** value equal to  
6582 ‘Seller.’ So we assume we have narrowed our focus to **CollaborationRole** node sets in Company  
6583 A’s *CPP* with **Role/@name='Buyer'** and in Company B’s **CollaborationRole** node sets with  
6584 **Role/@name='Seller'**.  
6585

6586 For more general collaborations, such as in the **MultiPartyCollaborations** of ebBPSS, we would  
6587 need to know the list of roles available within the process, and keep track of that for each of the  
6588 **CollaborationRoles**, the **Role** values chosen correspond correctly for the participants. We do not  
6589 here discuss the matching/filtering task for collaborations involving more than two roles, as  
6590 multiparty *CPAs* are not within scope for version 2.0 of this specification.  
6591

## 6592 E.5 Implementation Matching Tasks

6593 After filtering the CollaborationRoles with the desired ProcessSpecification, we should find one  
6594 CollaborationRole in our own CPP where we play the Buyer role, and one CollaborationRole in  
6595 our intended collaborator Company B's CPP where it plays the Seller role.

6596

6597 Our next task is to locate the specific candidate *bindings* relevant to *CPA* formation. There are  
6598 bindings for Service and Actions. For initial simplicity, we consider detailed matching tasks as  
6599 they arise for a standard collaboration case involving a *Request* action, followed by a *Response*  
6600 action. For version 2.0 of this specification, most matching tasks will involve matching of  
6601 referenced components of the *CPP*'s ***ThisPartyActionBinding*** elements under  
6602 ***CollaborationRole/ServiceBinding/CanSend/*** and under  
6603 ***CollaborationRole/ServiceBinding/CanReceive/***.

6604

### 6605 E.5.1 Action Correspondence and Selecting Correlative PackageIds, and ChannelIds

6606 In *CPPs*, under each of the elements, ***CollaborationRole/ServiceBinding/CanSend/*** and  
6607 ***CollaborationRole/ServiceBinding/CanReceive/***, are lists of ***ThisPartyActionBindings***. For  
6608 *Request-Response* collaboration patterns, we are interested in matches:

6609

- 6610 1. in the bindings of the Requesting side's ***CanSend/ThisPartyActionBinding*** with the  
6611 Responding side's ***CanReceive/ThisPartyActionBinding*** for the request action, and
- 6612 2. in the bindings of the Responding side's ***CanSend/ThisPartyActionBinding*** with the  
6613 Requesting side's ***CanReceive/ThisPartyActionBinding*** for the response action.

6614

6615 These correlative bindings give us references to detailed components that need to match for a  
6616 fully interoperable agreement. Case 1 pertains to the *Request*. Case 2 pertains to the *Response*.

6617

6618 For example, for Company A, we find under ***CanSend***:

6619

```
<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>
```

6620

6621 Correlative to this, for Company B, we find under ***CanReceive***:

6622

```
<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>
```

6623

6624 The correlation of elements can normally (when we are dealing with BPSS *Binary*

6641 Collaborations or their equivalents in other representations) be based on equality of the **action**  
6642 (or **requestOrResponseAction**) values. More detailed correlation of elements can make use of  
6643 more detailed testing and comparisons of the values in the **ActionContext** child elements of the  
6644 relevant **CanSend** and **CanReceive** pairs.

6645  
6646 In the preceding, we have illustrated the matching of **CanSend** and **CanReceive** for  
6647 asynchronous bindings. All **CanSend** bindings that are siblings under a **ServiceBinding** element  
6648 are asynchronous and make use of separate TCP connections that the **CanSend** side initiates on a  
6649 listening TCP port. In order to represent binding details for synchronous sending, the convention  
6650 is adopted whereby the **CanSend** element for a Receiver is placed under its **CanReceive** element.  
6651 This is illustrated by:

```
6652
6653 <tp:CanSend>
6654   <tp:ThisPartyActionBinding
6655     tp:id="companyA_ABID6"
6656     tp:action="Purchase Order Request Action"
6657     tp:packageId="CompanyA_RequestPackage">
6658       <tp:BusinessTransactionCharacteristics
6659         tp:isNonRepudiationRequired="true"
6660         tp:isNonRepudiationReceiptRequired="true"
6661         tp:isSecureTransportRequired="true"
6662         tp:isConfidential="transient"
6663         tp:isAuthenticated="persistent"
6664         tp:isTamperProof="persistent"
6665         tp:isAuthorizationRequired="true"
6666         tp:timeToAcknowledgeReceipt="PT2H"
6667         tp:timeToPerform="P1D"/>
6668       <tp:ActionContext
6669         tp:binaryCollaboration="Request Purchase Order"
6670         tp:businessTransactionActivity="Request Purchase Order"
6671         tp:requestOrResponseAction="Purchase Order Request Action"/>
6672       <tp:ChannelId>syncChannelA1</tp:ChannelId>
6673     </tp:ThisPartyActionBinding>
6674   <tp:CanReceive>
6675     <tp:ThisPartyActionBinding
6676       tp:id="companyA_ABID7"
6677       tp:action="Purchase Order Confirmation Action"
6678       tp:packageId="CompanyA_SyncReplyPackage">
6679         <tp:BusinessTransactionCharacteristics
6680           tp:isNonRepudiationRequired="true"
6681           tp:isNonRepudiationReceiptRequired="true"
6682           tp:isSecureTransportRequired="true"
6683           tp:isConfidential="transient"
6684           tp:isAuthenticated="persistent"
6685           tp:isTamperProof="persistent"
6686           tp:isAuthorizationRequired="true"
6687           tp:timeToAcknowledgeReceipt="PT2H"
6688           tp:timeToPerform="P1D"/>
6689         <tp:ActionContext
6690           tp:binaryCollaboration="Request Purchase Order"
6691           tp:businessTransactionActivity="Request Purchase Order"
6692           tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
6693         <tp:ChannelId>syncChannelA1</tp:ChannelId>
6694       </tp:ThisPartyActionBinding>
6695     </tp:CanReceive>
6696     <tp:CanReceive>
```

```
6697      <tp:ThisPartyActionBinding  
6698          tp:id="companyA_ABID8"  
6699          tp:action="Exception"  
6700          tp:packageId="CompanyA_ExceptionPackage">  
6701          <tp:BusinessTransactionCharacteristics  
6702              tp:isNonRepudiationRequired="true"  
6703              tp:isNonRepudiationReceiptRequired="true"  
6704              tp:isSecureTransportRequired="true"  
6705              tp:isConfidential="transient"  
6706              tp:isAuthenticated="persistent"  
6707              tp:isTamperProof="persistent"  
6708              tp:isAuthorizationRequired="true"  
6709              tp:timeToAcknowledgeReceipt="PT2H"  
6710              tp:timeToPerform="P1D"/>  
6711          <tp:ChannelId>syncChannelA1</tp:ChannelId>  
6712      </tp:ThisPartyActionBinding>  
6713      </tp:CanReceive>  
6714  </tp:CanSend>  
6715
```

6716 This subordination will also carry over to the synchronous receiving side, in which its  
6717 **CanReceive** element(s) is (are) under the **CanSend** element used to represent the initial sending  
6718 of a request. An illustration from Company B's synchronous binding is:

```
6719  
6720  <tp:CanReceive>  
6721      <tp:ThisPartyActionBinding  
6722          tp:id="companyB_ABID8"  
6723          tp:action="Purchase Order Request Action"  
6724          tp:packageId="CompanyB_SyncReplyPackage">  
6725          <tp:BusinessTransactionCharacteristics  
6726              tp:isNonRepudiationRequired="true"  
6727              tp:isNonRepudiationReceiptRequired="true"  
6728              tp:isSecureTransportRequired="true" tp:isConfidential="transient"  
6729              tp:isAuthenticated="persistent" tp:isTamperProof="persistent"  
6730              tp:isAuthorizationRequired="true" tp:timeToAcknowledgeReceipt="PT5M"  
6731              tp:timeToPerform="PT5M"/>  
6732      <tp:ActionContext  
6733          tp:binaryCollaboration="Request Purchase Order"  
6734          tp:businessTransactionActivity="Request Purchase Order"  
6735          tp:requestOrResponseAction="Purchase Order Request Action"/>  
6736          <tp:ChannelId>syncChannelB1</tp:ChannelId>  
6737      </tp:ThisPartyActionBinding>  
6738      <tp:CanSend>  
6739          <tp:ThisPartyActionBinding  
6740              tp:id="companyB_ABID6"  
6741              tp:action="Purchase Order Confirmation Action"  
6742              tp:packageId="CompanyB_ResponsePackage">  
6743              <tp:BusinessTransactionCharacteristics  
6744                  tp:isNonRepudiationRequired="true"  
6745                  tp:isNonRepudiationReceiptRequired="true"  
6746                  tp:isSecureTransportRequired="true"  
6747                  tp:isConfidential="transient"  
6748                  tp:isAuthenticated="persistent"  
6749                  tp:isTamperProof="persistent"  
6750                  tp:isAuthorizationRequired="true"  
6751                  tp:timeToAcknowledgeReceipt="PT5M"  
6752                  tp:timeToPerform="PT5M"/>  
6753          <tp:ActionContext
```

```

6754     tp:binaryCollaboration="Request Purchase Order"
6755     tp:businessTransactionActivity="Request Purchase Order"
6756     tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
6757     <tp:ChannelId>syncChannelB1</tp:ChannelId>
6758   </tp:ThisPartyActionBinding>
6759   </tp:CanSend>
6760   <tp:CanSend>
6761     <tp:ThisPartyActionBinding
6762       tp:id="companyB_ABID7"
6763       tp:action="Exception"
6764       tp:packageId="CompanyB_ExceptionPackage">
6765     <tp:BusinessTransactionCharacteristics
6766       tp:isNonRepudiationRequired="true"
6767       tp:isNonRepudiationReceiptRequired="true"
6768       tp:isSecureTransportRequired="true"
6769       tp:isConfidential="transient"
6770       tp:isAuthenticated="persistent"
6771       tp:isTamperProof="persistent"
6772       tp:isAuthorizationRequired="true"
6773       tp:timeToAcknowledgeReceipt="PT5M"
6774       tp:timeToPerform="PT5M"/>
6775     <tp:ChannelId>syncChannelB1</tp:ChannelId>
6776     </tp:ThisPartyActionBinding>
6777   </tp:CanSend>
6778 </tp:CanReceive>
6779

```

### 6780 E.5.2 Matching and Checking DeliveryChannel Details

6781 Until now, most of the matching work has been undertaken to find pairs of correlative action  
 6782 binding, and so the matching has functioned as a filtering mechanism. Once in possession of  
 6783 pairs of correlative action bindings, however, the work of checking for matches across the  
 6784 various dimensions of operation—transport, transport security, PKI compatibility for various  
 6785 tasks, agreement about messaging characteristics (reliable messaging, digital enveloping, signed  
 6786 acknowledgments (minimal non-repudiation of receipt), non-repudiation of origin, packaging  
 6787 details, and more begins.

6788 Once in possession of the action bindings, IDREFs provide references to the underlying  
 6789 components for comparison. For example, when comparing packaging details, the *Request*  
 6790 IDREFS are found at ***CanSend/ThisPartyActionBinding/@packageId*** and within the other *CPP*  
 6791 at ***CanReceive/ThisPartyActionBinding@packageId***. For Company A's *Request "Purchase*  
 6792 *Order Request Action,"* the packaging IDREF is found in:

6793     tp:packageId="CompanyA\_RequestPackage"

6794 and this IDREF value refers to:

```

6795 <tp:Packaging tp:id="CompanyA_RequestPackage">
6796   <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>
6797   <tp:CompositeList>
6798     <tp:Composite tp:id="CompanyA_RequestMsg"
6799       tp:mimetype="multipart/related" tp:mimeparameters="type=text/xml;">
6800       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
6801       <tp:Constituent tp:idref="CompanyA_Request"/>
6802     </tp:Composite>
6803
6804
6805
6806

```

6807           </tp:CompositeList>  
6808        </tp:Packaging>  
6809

6810 For Company A's *Request "Purchase Order Request Action"*, the delivery channel IDREF is  
6811 found in:

6812  
6813   <tp:ChannelId>asyncChannelA1</tp:ChannelId>  
6814

6815 and this IDREF value refers to the element with this ID, namely:

6816  
6817   <tp:DeliveryChannel tp:channelId="asyncChannelA1"  
6818     tp:transportId="transportA1" tp:docExchangeId="docExchangeA1">  
6819     <tp:MessagingCharacteristics  
6820        tp:syncReplyMode="none"  
6821        tp:ackRequested="always"  
6822        tp:ackSignatureRequested="always"  
6823        tp:duplicateElimination="always"/>  
6824    </tp:DeliveryChannel>  
6825

6826 Two remaining crucial references for understanding the binding, are found in attributes of the  
6827 *DeliveryChannel*, namely: *DeliveryChannel/@transportId* and in the attribute  
6828 *DeliveryChannel/@docExchangeId*.

6829  
6830 For Company A, for example, we find *transportId*="transportA1" and  
6831 *docExchangeId*="docExchangeA1" are the IDREFs for the continuing binding information with  
6832 the *DeliveryChannel*, "asyncChannelA1". Resolving these references, we obtain:

6833  
6834   <tp:Transport tp:transportId="transportA1">  
6835     <tp:TransportSender>  
6836       <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>  
6837       <tp:TransportClientSecurity>  
6838         <tp:TransportSecurityProtocol  
6839           tp:version="3.0">SSL</tp:TransportSecurityProtocol>  
6840           <ClientCertificateRef tp:certId="CompanyA\_ClientCert"/>  
6841             <tp:ServerSecurityDetailsRef  
6842               tp:securityId="CompanyA\_TransportSecurity"/>  
6843             </tp:TransportClientSecurity>  
6844           </tp:TransportSender>  
6845           <tp:TransportReceiver>  
6846            <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>  
6847            <tp:Endpoint  
6848              tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/async"  
6849              tp:type="allPurpose"/>  
6850            <tp:TransportServerSecurity>  
6851            <tp:TransportSecurityProtocol  
6852              tp:version="3.0">SSL</tp:TransportSecurityProtocol>  
6853              <tp:ServerCertificateRef tp:certId="CompanyA\_ServerCert"/>  
6854                <tp:ClientSecurityDetailsRef  
6855                tp:securityId="CompanyA\_TransportSecurity"/>  
6856                </tp:TransportServerSecurity>  
6857                </tp:TransportReceiver>  
6858        </tp:Transport>  
6859

6860 for *transportID* "transportA1" and

```
6861
6862 <tp:DocExchange tp:docExchangeId="docExchangeA1">
6863     <tp:ebXMLSenderBinding tp:version="2.0">
6864         <tp:ReliableMessaging>
6865             <tp:Retries>3</tp:Retries>
6866             <tp:RetryInterval>PT2H</tp:RetryInterval>
6867             <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
6868         </tp:ReliableMessaging>
6869         <tp:PersistDuration>P1D</tp:PersistDuration>
6870         <tp:SenderNonRepudiation>
6871             <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#
6872         </tp:NonRepudiationProtocol>
6873         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1
6874         </tp:HashFunction>
6875         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-sha1
6876     </tp:SignatureAlgorithm>
6877     <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
6878     </tp:SenderNonRepudiation>
6879     <tp:SenderDigitalEnvelope>
6880     <tp:DigitalEnvelopeProtocol
6881         tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
6882         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
6883         <tp:EncryptionSecurityDetailsRef
6884             tp:securityId="CompanyA_MessageSecurity"/>
6885         </tp:SenderDigitalEnvelope>
6886     </tp:ebXMLSenderBinding>
6887     <tp:ebXMLReceiverBinding tp:version="2.0">
6888         <tp:ReliableMessaging>
6889         <tp:Retries>3</tp:Retries>
6890         <tp:RetryInterval>PT2H</tp:RetryInterval>
6891         <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
6892     </tp:ReliableMessaging>
6893     <tp:PersistDuration>P1D</tp:PersistDuration>
6894     <tp:ReceiverNonRepudiation>
6895         <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#
6896         </tp:NonRepudiationProtocol>
6897         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1
6898         </tp:HashFunction>
6899         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-sha1
6900     </tp:SignatureAlgorithm>
6901     <tp:SigningSecurityDetailsRef
6902         tp:securityId="CompanyA_MessageSecurity"/>
6903     </tp:ReceiverNonRepudiation>
6904     <tp:ReceiverDigitalEnvelope>
6905     <tp:DigitalEnvelopeProtocol
6906         tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
6907         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
6908         <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
6909     </tp:ReceiverDigitalEnvelope>
6910     </tp:ebXMLReceiverBinding>
6911 </tp:DocExchange>
```

6912 for the ***docExchangeId***, docExchangeA1.

6913

6914 There are, of course, other references, such as those to security-related capabilities, that will be

6916 important to resolve when checking detailed matching properties, but the four IDREFs (two for  
6917 the sender and two for the receiver) that have just been introduced are critical to the remainder of  
6918 the match tests that will lead to the formation of draft *CPAs*. We will assume at this point that the  
6919 reader can resolve IDREFs using the example *CPPs* and *CPAs* for Company A and B in the  
6920 appendices, and will not exhibit them in the text in order to save space.

6921  
6922 We next turn to a more in-depth treatment of the tests that are involved in finding the elements  
6923 for a draft *CPA*.

6924  
6925 The detailed tasks to be discussed in greater depth are:

- 6926
- 6927 1. Matching *Channel MessagingCharacteristics*
  - 6928 2. Checking *BusinessTransactionCharacteristics* coherence with *Channel* details
  - 6929 3. Matching *Packaging*
  - 6930 4. Matching *Transport* and *Transport[Receiver|Sender]Security*
  - 6931 5. Matching and Checking *DocExchange* subtrees.

6932  
6933 Because agreement about *Transport* is quite fundamental, we shall consider it first.  
6934 Computational processes are likely to first find pairs that match on *Transport* details, and will  
6935 ignore pairs failing to have matches at this level.

#### 6936 E.5.2.1 Matching Transport

6937 Matching *Transport* first involves matching the *Transport/TransportSender/TransportProtocol*  
6938 capabilities of the requester with the *Transport/TransportReceiver/TransportProtocol*  
6939 capabilities found under the collaborator receiving the *request*. Several such matches can exist,  
6940 and any of these matches can be used in forming a draft, provided other aspects match up  
6941 satisfactorily. Each *CPP* is assumed to have listed its preferred transport protocols first (as  
6942 determined by the listing of the Bindings that reference the *Transport* element, but different  
6943 outcomes can result depending on which *CPP* is used first for searching for matches. In general,  
6944 resolution of preference differences is left to a distinct phase of *CPA negotiation*, following  
6945 proposal of a draft *CPA*. Negotiation can be performed by explicit actions of users, but is  
6946 expected to become increasingly automated.

6947  
6948 Matching transport secondly involves matching the *TransportSender/TransportProtocol*  
6949 capabilities of the responding collaborator with its *TransportReceiver/TransportProtocol*  
6950 capabilities found under the collaborator receiving the response, which is typically the  
6951 collaborator that has sent a request. Several such matches can exist, and any of these matches can  
6952 be used in forming a draft. In one case, however, there may be no need for the second match on  
6953 *TransportProtocol*. If we are using HTTP or some other protocol supporting synchronous  
6954 replies, and the *DeliveryChannel* has a *MessagingCharacteristics* child that has its  
6955 *syncReplyMode* attribute with a value of “signalsAndResponse,” then everything comes back  
6956 synchronously, and there is no need to match on *TransportProtocol* for the *Response*  
6957 *DeliveryChannel*.

6958  
6959 If *TransportSecurity* is present, then there can be additional checks. First,  
6960 *TransportSender/TransportClientSecurity/TransportSecurityProtocol* should be compatible

6962 with **TransportReceiver/TransportServerSecurity/TransportSecurityProtocol**. Second, if either  
6963 the **TransportSender/TransportClientSecurity/ClientSecurityDetailsRef** or  
6964 **TransportSender/TransportClientSecurity/ServerSecurityDetailsRef** elements are present, and  
6965 the IDREF references an element containing some **AnchorCertificateRef**, then an opportunity  
6966 exists to check suitability of one *Party*'s PKI trust of the certificates used in the  
6967 **TransportSecurityProtocol**. For example, by resolving the IDREF value in  
6968 **TransportSender/TransportClientSecurity/ClientCertificateRef@certId**, we can obtain the  
6969 proposed client certificate to use for client-side authentication. By resolving the IDREFs from  
6970 the **AnchorCertificateRef**, we become able to determine whether the proposed client certificate  
6971 will "chain to a trusted root" on the server side's PKI. Similar remarks apply to checks on the  
6972 validity of a server certificate found by resolving  
6973 **TransportReceiver/TransportServerSecurity/ServerCertificateRef**. This server certificate can  
6974 be checked against the CA trust anchors that are found by resolving  
6975 **TransportSender/TransportClientSecurity/ServerSecurityDetailsRef@securityId**, and finding  
6976 CA certificates (or CA certificate chains) in the **KeyInfo** elements under the **Certificate** element  
6977 obtained by resolving the IDREF found in **AnchorCertificateRef@certId**.

6979 When matches exist for the correlative **Transport** components, we then have discovered an  
6980 interoperable solution at the transport level. If not, no *CPA* will be available, and a gap has been  
6981 identified that will need to be remedied by whatever exception handling procedures are in place.  
6982 Let us next consider other capabilities that need to match for "thicker" interoperable solutions.

#### 6984 E.5.2.2 Checking BusinessTransactionCharacteristics and DeliveryChannel 6985     MessagingCharacteristics

6986 Under each of the correlative action bindings, there is a child element of **DeliveryChannel**,  
6987 **MessagingCharacteristics** that has several attributes important in *CPA* formation tasks. The  
6988 attributes having wider implications are **syncReplyMode**, **ackRequested**, and  
6989 **ackSignatureRequested**; for the **duplicateElimination** and **actor** attributes, compatibility exists  
6990 when the attributes that are found under the **CanSend** and **CanReceive DeliveryChannels** have  
6991 the same values. As the element's name implies, all of these **DeliveryChannel** features pertain to  
6992 the messaging layer.

6993 In addition, **BusinessTransactionCharacteristics**, found under **ThisPartyActionBinding**,  
6994 contains attributes reflecting a variety of features pertaining to desired security and business  
6995 transaction properties that are to be implemented by the agreed upon **DeliveryChannels**. These  
6996 properties may have implications on what capabilities are needed within more detailed  
6997 components of the **DeliveryChannel** elements, such as in the **Packaging** element. When using a  
6998 *BPSS* process specification, these properties may be specified within the *BusinessTransaction*.  
6999 The properties of the **BusinessTransactionCharacteristics** element are, however, the ones that  
7000 will be operative in the implementation of the **BusinessTransaction**, and may override the  
7001 specified values found in the *BPSS* Process specification. Because the properties are diverse, the  
7002 details that implement the properties can be spread over other elements referenced within the  
7003 **DeliveryChannel** elements.

7004 These attributes apply to either a *Request* or *Response* delivery channel, but can impact either the  
7005 *Sender* or *Receiver* (or both) in a channel. In addition, the attributes governing

7008 acknowledgments, for example, qualify the interrelation of ***DeliveryChannel*** elements by  
7009 specifying behavior that is to occur that qualifies the contents of a return message.  
7010  
7011 The most basic test for compatibility for any of the attributes in either ***MessagingCharacteristics***  
7012 or ***BusinessTransactionCharacteristics*** is that the attributes are equal in the sending party's  
7013 ***DeliveryChannel*** referenced by ***CanSend/ThisPartyActionBinding/ChannelId*** and in the  
7014 receiving party's ***DeliveryChannel*** referenced by  
7015 ***CanReceive/ThisPartyActionBinding/ChannelId***. If they are unequal, and all Bindings have  
7016 been examined on both sides, a draft *CPA* will represent a compromise to some common set with  
7017 respect to the functionality represented by the attributes.

7018  
7019 In the following discussions, we will consider many of the attributes in the two *Characteristics*  
7020 elements, and relate them to additional underlying implementational details, one of which is  
7021 ***Packaging***.  
7022

7023 From a high level, basic agreement in *packaging* is a matter of compatibility of the generated  
7024 *packaging* on the sending side with the parsed packaging on the receiving side. The basic  
7025 packaging check is, therefore, checking packaging compatibility under the ***CanSend*** element of a  
7026 sender action with the packaging under the ***CanReceive*** element of that same action under the  
7027 receiver side.  
7028

7029 For efficiency, representation of capabilities of parsing/handling packaging can make use of both  
7030 wildcards and repetition, and as needed these capabilities can also express open data formatting  
7031 used on the generating side. For example, consider the ***SimplePart***:

7032       <tp:SimplePart tp:id="IWild" tp:mimetype="\*/\*"/>  
7033

7034 By wildcarding ***mimetype*** values, we represent our capability of accepting any data, and would  
7035 match any specific MIME type. Also, consider a ***Constituent*** appearing within a ***Composite***:

7036       <tp:Constituent tp:idref="MsgHdr"/>  
7037       <tp:Constituent minOccurs="0" maxOccurs="10" tp:idref="IWild"/>  
7038

7039 This notation serves to capture the capability of handling any number of arbitrary MIME  
7040 bodyparts within the ***Composite*** being defined. A Packaging capability such as this would  
7041 obviously match numerous more specific generated *Packaging* schemes, as well as matching  
7042 literally with a scheme of the same generality.  
7043

7044 Certain more complex checks are needed for more complicated packaging options pertaining to  
7045 syncReplyMode. These are discussed in the following.  
7046

#### 7047 ***syncReplyMode***

7048 The ***syncReplyMode*** has a value other than "none" to indicate what parts of a message should be  
7049 returned in the *Reply* of a transport capable of synchronous operation, such as HTTP. (We here  
7050 use "synchronous" to mean "on the same TCP connection," which is one use of this term. We do  
7051 not specify any waiting, notification, or blocking behavior on processes or threads that are  
7052 involved, though presumably there is some computational activity that maintains the connection  
7053  
7054

7055 state and is above the TCP and socket layers.)

7056  
7057 The possible implementations pertaining to various values of the *syncReplyModes* are numerous,  
7058 but we will try to indicate at least the main factors that are involved.

7059  
7060 As will be seen, the **Packaging** element is important in specifying implementation details and  
7061 compatibilities. But, because business level signals may be involved, other action bindings may  
7062 need examination in addition to the already selected bindings for the *Request* and *Response*.  
7063 Also, the values of **TransportReceiver/Endpoint/@type** might need checking when producing  
7064 draft *CPAs*.

7065  
7066 Let us first begin with the cases in which *Responses*, *Message Service Handler Signals* and  
7067 *Business Signals* return in some combination of a synchronous reply and other asynchronous  
7068 message(s). These various combinations will be discussed for the *syncReplyMode* values:  
7069 "mshSignalsOnly," "signalsOnly," "responseOnly," and "signalsAndResponse."

7070  
7071 By convention, synchronous replies are represented by subordinating **CanSend** or **CanReceive**  
7072 elements under the **CanReceive** or **CanSend** elements that represent the initial *Request* binding  
7073 capabilities. For representing asynchronous Requests, Replies, or Signals, the **CanSend** or  
7074 **CanReceive** elements are all siblings and directly subordinate to the **ServiceBinding**. Therefore,  
7075 both asynchronous and synchronous capabilities can be grouped under a **ServiceBinding** in a  
7076 *CPP*, and can still be unambiguously distinguished. In principle, increasing subordination  
7077 (nesting) can indicate patterns of dialog more elaborate than *Request* and *Response*. Few use  
7078 cases for this functionality are common at the time of this writing.

7079  
7080 **mshSignalsOnly**

7081 The Request sender's **DeliveryChannel** (referenced by  
7082 **CanSend/ThisPartyActionBinding/ChannelId**) and the Request receiver's **DeliveryChannel**  
7083 (referenced by **CanReceive/ThisPartyActionBinding/ChannelId**) both should have  
7084 **MessagingCharacteristics/@syncReplyMode** value of **mshSignalsOnly**.

7085  
7086 While a Party can explicitly identify a **DeliveryChannel** for the SOAP envelope with subordinate  
7087 **CanSend** and **CanReceive** elements, and with them specialized bindings, these are typically  
7088 omitted for ebXML Messaging software. It is presumed that each side can process a synchronous  
7089 reply constructed in accordance with ebXML Messaging. The **DeliveryChannel** representation  
7090 mechanism here serves as a placeholder for capturing other Messaging Signal protocols that  
7091 might emerge.

7092  
7093 Currently acknowledgments and signed acknowledgments, along with errors, are the primary  
7094 MSH signals that are included in the SOAP envelope. If Company A set  
7095 **syncReplyMode** to **mshSignalsOnly**, then Company B's correlative  
7096 **CanReceive/ThisPartyActionBinding/@packageId** should contain a nested  
7097 **CanSend/ThisPartyActionBinding/@packageId** for a message without any business payload or  
7098 signals. In addition, the **CanSend/ThisPartyActionBinding/@packageId** of Company B's  
7099 *Response* should resolve to packaging format capable of returning the *Response* ( and possibly  
7100 other constituents) asynchronously. The compatibility of the **DeliveryChannel** elements can be

7101 checked, as can the capability of Company A to receive that Response payload, the Signal  
7102 payload(s), or Responses bundled with signals as specified by the packaging formats that are  
7103 referenced through the relevant **ThisPartyActionBindings** element's *packageId* attribute values.  
7104

7105 **signalsOnly**

7106 The Request sender's **DeliveryChannel** (referenced by its  
7107 **CanSend/ThisPartyActionBinding/ChannelId**) and the Request receiver's **DeliveryChannel**  
7108 (referenced by its **CanReceive/ThisPartyActionBinding/ChannelId**) both should have  
7109 **MessagingCharacteristics/@syncReplyMode** value of **signalsOnly**.  
7110

7111 If Company A sets **syncReplyMode** to **signalsOnly**, then under Company B's correlative  
7112 **CanReceive** element, there should be a nested **CanSend/ThisPartyActionBinding** whose  
7113 *packageId* attribute's value resolves to a packaging format appropriate for Signals. For the  
7114 **CanSend/ThisPartyActionBinding/@packageId** associated with Company B's business level  
7115 **Response**, the attribute IDREF value should resolve to a packaging format capable of returning  
7116 payloads and that omits business signals. This **CanSend** element will be a direct child of  
7117 **ServiceBinding**, a placement representing its asynchronous character. The original requesting  
7118 party will need to have a **CanReceive/ThisPartyActionBinding** that is compatible with the  
7119 responding party, and that is a direct child of its **ServiceBinding** element.  
7120

7121 Using subordinate **CanSend** and subordinate **CanReceive** elements can be useful if the  
7122 **DeliveryChannel** details for Exception signals differ from those specified for Request and  
7123 Response. Signal bindings, for example, may differ by omitting **ackRequested**, or possibly one  
7124 of the security features (digital enveloping or non-repudiation of receipt) that are used for  
7125 Requests or Responses. Just as with other tests on Requests and Responses, there can be checks  
7126 for compatibility in **Packaging**, **DocExchange**, **MessagingCharacteristics**, or  
7127 **BusinessTransactionCharacteristics** referred to in the correlative subordinate **CanSend** and  
7128 **CanReceive DeliveryChannels**.  
7129

7130 **responseOnly**

7131 The Request sender's **DeliveryChannel** (referenced by  
7132 **CanSend/ThisPartyActionBinding/ChannelId**) and the Request receiver's **DeliveryChannel**  
7133 (referenced by **CanReceive/ThisPartyActionBinding/ChannelId**) both should have  
7134 **MessagingCharacteristics/@syncReplyMode** value of **responseOnly**.  
7135

7136 If Company A sets **syncReplyMode** to **responseOnly**, the  
7137 **CanSend/ThisPartyActionBinding/@packageId** of Company B's response should resolve to a  
7138 packaging format capable of returning payloads, but omitting business signals. The  
7139 **CanSend/ThisPartyActionBinding** element will be included as a child of the **CanReceive**  
7140 element so the responder can indicate that it is a synchronous response.  
7141

7142 There should be an independent way to return business level error signals. So, there should be a  
7143 **ThisPartyActionBinding** for any Signal payload announced, and these bindings should be at the  
7144 direct child of **ServiceBinding** level to represent their asynchronous flavor.  
7145

7146 It is not too likely that **ReceiptAcknowledgment** and similar signals will be used when a response

7147 is returned synchronously. The motivation for using these signals is indicating positive forward  
7148 progress, and this motivation will be undermined when a Response is returned directly.  
7149

7150 For the **responseOnly** case, including subordinate **CanSend/ThisPartyActionBinding** and  
7151 **CanReceive/ThisPartyActionBinding**, means that there can be checks for compatibility in  
7152 **Packaging**, **DocExchange**, **MessagingCharacteristics**, or **BusinessTransactionCharacteristics**.  
7153 The **syncReplyMode** and **ackRequested** attributes here should be carefully considered because a  
7154 **mshSignalsOnly** value here would mean that another round of synchronous messaging will need  
7155 to occur on the same connection. Incidentally, for **Transport** elements referenced under  
7156 subordinate bindings, there need not be any **Endpoint** elements. If there are **Endpoint** elements,  
7157 they may be ignored.  
7158

#### 7159 **signalsAndResponse**

7160 The Request sender's **DeliveryChannel** (referenced by  
7161 **CanSend/ThisPartyActionBinding/ChannelId**) and the Request receiver's **DeliveryChannel**  
7162 (referenced by **CanReceive/ThisPartyActionBinding/ChannelId**) both should have  
7163 **MessagingCharacteristics/@syncReplyMode** value of **signalsAndResponse**.  
7164

7165 If Company A sets **syncReplyMode** to **signalsAndResponse**, the  
7166 **CanSend/ThisPartyActionBinding** of Company B's response should be subordinate to Company  
7167 B's **CanReceive** element. The packaging format that is referenced should be capable of  
7168 returning payloads and signals bundled together. If no asynchronous bindings exist for error  
7169 signals, this will be the only defined **DeliveryChannel** agreed to for all aspects of message  
7170 exchange for the business transaction. However, it is likely that an asynchronous binding would  
7171 normally be provided to send Exception signals.  
7172

#### 7173 **ackRequested and ackSignatureRequested**

7174 Checks on the **ackRequested** and **ackSignatureRequested** attributes within correlative  
7175 **DeliveryChannels** (that is, correlative because referenced under one action's **CanSend** and  
7176 **CanReceive** elements) are primarily to see that the values of the corresponding attributes are the  
7177 same.  
7178

7179 However, there are some interactions of these attributes with other information items that need to  
7180 be mentioned.  
7181

7182 The principal use of the **ackRequested** attribute is within reliable messaging configurations. If  
7183 reliable messaging is to be configured, then checks on agreement in the correlative  
7184 **ReliableMessaging** elements as found under **DocExchange/ebXMLSenderBinding** and  
7185 **DocExchange/ebXMLReceiverBinding** are in order. Also, the value of the  
7186 **duplicateElimination** attribute of **MessagingCharacteristics** should be checked for agreement.  
7187 Draft CPAs may be formed by deliberately aligning values that are not equal along some of these  
7188 dimensions. Downgrading may provide draft CPAs most likely to gain acceptance; so, for  
7189 example, if **duplicateElimination** is false on the receiving side, aligning it to false on the sending  
7190 side is most likely to produce a draft that succeeds.  
7191

7192 The additional function of **ackSignatureRequested** is that it provides a "thin" implementation for

7193 *non-repudiation of receipt*. The basic check is for equality of attribute value, but additional  
7194 constraints may need test and alignment. If no signal capable of implementing *non-repudiation*  
7195 of receipt is found under the **ServiceBinding**, then having an “always” value for  
7196 **ackSignatureRequested** suggests aligning the **BusinessTransactionCharacteristics** attributes,  
7197 **isNonRepudiationReceiptRequired**, to be true. However, if this is done, care should be taken to  
7198 check that the **BusinessTransactionCharacteristics** attribute **isIntelligibleCheckRequired** is  
7199 false. This is because the messaging implementation only deals with receipt in the sense of  
7200 having received a byte stream off the wire (and persisting it so that it is available for further  
7201 processing). It is not safe to presume that any syntactical or semantic checks on the data were  
7202 performed.

7203  
7204 **E.5.2.3 DocExchange Checks for BusinessTransactionCharacteristics**  
7205 When using *CPPs* and *CPAs* with ebXML Messaging, which is the most likely early deployment  
7206 situation, there exists an opportunity to check agreement on **BusinessTransactionCharacteristics**  
7207 attributes:

7208  
7209 The following three attributes need to have equal values in the bindings for a Request or for a  
7210 Response. No further discussion will be provided in this appendix on these “deadlines,” except to  
7211 say that a sophisticated proposed *CPA* generation tool might check on the coherence of the  
7212 values chosen here with values for reliable messaging parameters, existence of compatible  
7213 ReceiptAcknowledgment or AcceptanceAcknowledgment bindings, and consistency with  
7214 syncReplyMode internal configuration.

7215  
7216 <attribute name="timeToAcknowledgeReceipt" type="duration"/>  
7217 <attribute name="timeToAcknowledgeAcceptance" type="duration"/>  
7218 <attribute name="timeToPerform" type="duration"/>

7219  
7220 The remaining attributes involve a number of security related issues and will be the focus of the  
7221 remaining discussion of **BusinessTransactionCharacteristics** attributes:

7222  
7223 <attribute name="isNonRepudiationRequired" type="boolean"/>  
7224 <attribute name="isNonRepudiationReceiptRequired" type="boolean"/>  
7225 <attribute name="isIntelligibleCheckRequired" type="boolean"/>  
7226 <attribute name="isAuthenticated" type="tns:persistenceLevel.type"/>  
7227 <attribute name="isTamperProof" type="tns:persistenceLevel.type"/>  
7228 <attribute name="isAuthorizationRequired" type="boolean"/>  
7229 <attribute name="isConfidential" type="tns:persistenceLevel.type"/>  
7230 <attribute name="isSecureTransportRequired" type="boolean"/>

7231  
7232 Here, the basic test is that for correlative **DeliveryChannels**, the corresponding attributes have  
7233 the same values. Again there are some interaction aspects with parts of the **DeliveryChannel** that  
7234 motivate making some additional checks.

7235  
7236 Previously, when discussing the **MessagingCharacteristics** attribute **ackSignatureRequested**, it  
7237 was pointed out that the messaging implementation provides thin support for holding  
7238 **isNonRepudiationReceiptRequired** true provided that the attribute **isIntelligibleCheckRequired**  
7239 is false. When both are true, then there should exist a business signal with compatible **Packaging**  
7240 and **DeliveryChannel** values. If the signal has been independently described within  
7241 asynchronous **CanSend** and **CanReceive** elements, knowing the signal name (such as,  
7242 “ReceiptAcknowledgment”) may support a relatively simple search and test. However, if

7243 synchronous transports are involved, some filters using *syncReplyModes* may be needed to  
7244 discover an underlying support for a “thick” implementation of *non-repudiation of receipt*.  
7245

7246 When non-repudiation of receipt is implemented by a business signal, then checks on signing  
7247 certificate validity can involve the *CollaborationRole/ApplicationCertificateRef* and the  
7248 *CollaborationRole/ApplicationSecurityDetailsRef*, that provides a reference to the  
7249 *SecurityDetails* element containing the list of *TrustAnchors*. The certificate from the side  
7250 signing the ReceiptAcknowledgment would be checked against the certificates referred to by the  
7251 *AnchorCertificateRef* under *TrustAnchors*.

7252

7253 The business signal will sometimes be conveyed as part of a message. It remains true that the  
7254 message itself will still be sent through a MSH, and that the MSH can also sign the message  
7255 using the certificate found by resolving the IDREF found at  
7256 *DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId*.  
7257

7258 If a particular software component implements both MSH functionality and business level  
7259 security functionality, it is possible that the same certificate may be pointed to by  
7260 *ApplicationCertificateRef* and *SigningCertificateRef/@certId*. In other words, the distinction  
7261 between MSH level signing and application level signing is a logical one, and may not  
7262 correspond with software component boundaries. Because the MSH signature is over the  
7263 message, the message signature may be over an application level signature. While this may be  
7264 redundant for some system configurations, protocols may require both signatures to exist over  
7265 the different regions.

7266

7267 Failure to validate a certificate may not prevent formation of a draft *CPA*. First, the sender’s signing  
7268 certificate can be a self-signed certificate. If so, a reference to this self-signed certificate may be  
7269 added to the receiver’s *TrustAnchors/AnchorCertificateRef* list. This proposal amounts to  
7270 proposing to agree to a direct trust model, rather than a hierarchical model involving certificate  
7271 authorities. Second, a proposal to add a trusted root may be made, again by appropriate revision of  
7272 the *TrustAnchors*.

7273

7274 When non-repudiation of receipt is implemented by the Messaging layer, the checks on PKI  
7275 make use of elements under *DocExchange*.

7276

7277 *isNonRepudiationRequired*  
7278 *isAuthenticated*  
7279 *isAuthorizationRequired*  
7280 *isTamperProof*

7281

7282 The ideas of authentication, authorization, nonrepudiation and being “tamper proof” may be very  
7283 distinct as business level concepts, yet the implementation of these factors tend to use very  
7284 similar technologies. Actually, prevention of tampering is not literally implemented. Instead,  
7285 means are provided for detecting that tampering (or some accidental garbling) has occurred.  
7286 Likewise, implementations of authorization usually are provided by implementations of access  
7287 control (permitting or prohibiting a user in a role making use of a resource) and presentation of a  
7288 token or credential to gain access, which may involve authentication as an initial step!

7289 Nonrepudiation may build on all the previous functions, plus retaining information for supplying  
7290 presumptive evidence of origination at some later time.

7291  
7292 When checking whether ***isNonRepudiationRequired*** can be set to True for both parties, check  
7293 whether the signing certificate will be counted as valid at the receiver.

7294 The IDREF reference to the signing certificate is found in

7295 ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId***.  
7296 The referenced certificate should be checked for validity with respect to the trust anchors  
7297 obtained from ***TrustAnchors/AnchorCertificateRef*** elements under the ***SecurityDetails***  
7298 element referenced by the IDREF at

7299 ***DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation/SigningSecurityDetailsRef/***  
7300 ***@securityId***.

7301  
7302 As previously noted, failure to validate a certificate does not prevent constructing a draft CPA.  
7303 Either self-signed certificates or new trust anchors can be added to align the trust model on one  
7304 side with the other side's certificate.

7305  
7306 In addition to checking the interoperability of the PKI infrastructures, checks on compatibility of  
7307 values in the other attributes in

7308 ***DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation*** and in  
7309 ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation*** can be made.

7310 ***NonRepudiationProtocol***, ***HashFunction***, and ***SignatureAlgorithm*** values may be compatible  
7311 even when not equal if knowledge of the protocol requirements allows fallback to a mandatory to  
7312 implement value. So values here can be found equal, aligned, or negotiated to reach an  
7313 agreement.

7314  
7315 If ***isNonRepudiationRequired*** is True, the ***isAuthenticated*** and ***isTamperProof*** should also be  
7316 True. This is because in implementing ***isNonRepudiationRequired*** by means of a digital  
7317 signature, both authentication (with respect to the identity associated with the signing certificate)  
7318 and tamper detection (with respect to the cryptographic hash of the signature) will be  
7319 implemented as well. The converses need not be true because authentication and tamper  
7320 detection might be accomplished without archiving information needed to support claims of  
7321 nonrepudiation.

7322  
7323 ***isConfidential***

7324 ***isSecureTransportRequired***

7325

7326 The ***isConfidential*** and ***isSecureTransportRequired*** attributes indicate properties variously  
7327 distributed among levels of the application-to-application sending/receiving stacks.

7328 ***isConfidential*** has possible values of "none", "transient", "persistent", and "transient-and-  
7329 persistent". If ***isSecureTransportRequired*** is true, there should be compatible  
7330 ***TransportSecurity***, and ***isConfidential*** needs to have either a "transient" or a "transient-and-  
7331 persistent" value for consistency. The "persistent" or "transient-and-persistent" values indicate  
7332 that some digital enveloping function is present.

7333

7334 ebXML Messaging version 2.0 does not have an "official" implementation for digital envelopes,

7335 and refers to the future XML Encryption specification as its intended direction for that function.  
7336 However, the XML Encryption specification is now a candidate recommendation, and is suitable  
7337 for preliminary implementation.

7338  
7339 Within the *CPA*, the *DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope* and  
7340 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope* can provide configuration  
7341 details pertaining to security in accordance with [XMLENC]. Use of XML Encryption also will  
7342 normally show up in the value of *DigitalEnvelopeProtocol*, and can also appear within a  
7343 *NamespaceSupported* element within *Packaging*.

7344  
7345 Currently, [ebMS] has only indicated a direction to eventually use XML Encryption, but has not  
7346 mandated any digital envelope protocol. Digital enveloping may be done at the “application  
7347 level,” and will show up under MIME types within the *Packaging* element. PKI matching will  
7348 make use of certificates supplied in *ApplicationCertificateRef* and  
7349 *ApplicationSecurityDetailsRef*. If other protocols are to be used, it would be safest to use  
7350 extensions to the content model of *DocExchange*, such as, *XXXSenderBinding* and  
7351 *XXXReceiverBinding*, and follow the pattern of the ebXML content models for *DocExchange*.  
7352 Future versions of this specification intend to make these extension semantics easier to use  
7353 interoperably; currently, the extensions would be a multilateral extension within some trading  
7354 community.

7355  
7356 When checking whether *isConfidential* can be set to “persistent” or “transient-and-persistent”  
7357 for both parties, check whether the key exchange certificate will be counted as valid at the  
7358 sender. The IDREF reference to the *SecurityDetails* element is found in  
7359 *DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope/EncryptionSecurityDetailsRef/@*  
7360 *securityId*. The trust anchor certificates obtained from *TrustAnchors/AnchorCertificateRef*  
7361 elements under the *SecurityDetails* element will be used to test that the certificate referenced by  
7362 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@c*  
7363 *ertId* validates at the sender side.

7364  
7365 As previously noted, failure to validate a certificate does not prevent constructing a draft CPA.  
7366 Either self-signed certificates or new trust anchors can be added to align the trust model on one  
7367 side with the other side’s certificate.

7368  
7369 In addition to the PKI related checks and alignments, the elements *EncryptionAlgorithm* and  
7370 *DigitalEnvelopeProtocol* should be checked for equality (or compatibility) and, if not  
7371 compatible or equal, aligned to values that would work for an initial version of a proposed *CPA*.  
7372 Preferences and alignment of these elements can be achieved in a subsequent Negotiation phase.

7373  
7374 Finally, it is possible that one side’s DigitalEnvelope will be modeled using either the  
7375 *DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope* and  
7376 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope*, while the other side uses only  
7377 *Packaging* to indicate use of, for example, S/MIME Digital Envelopes, because it receives an  
7378 already enveloped payload from an application. In such a case, the PKI certificate  
7379 validationcheck could require checking that a certificate described by  
7380 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@c*

7381     *ertId* validates against the ***TrustAnchors*** found by resolving  
7382     ***CollaborationRole/ApplicationSecurityDetailsRef***. This complication arises from the possibility  
7383     that digital enveloping functionality can be spread over quite distinct portions of the stack in  
7384     different software installations.

7385

## 7386     **E.6 CPA Formation: Technical Details**

7387     When assembling a draft *CPA* from matching portions of two *CPPs*' ***PartyInfo*** elements, some  
7388     additional constraints need to be observed.

7389

7390     First, as mentioned in section 9.11.1, software for producing draft CPAs needs to guarantee that  
7391     ID values in one *CPP* are distinct from ID values in the other *CPP* so that no IDREF references  
7392     collide when the *CPPs* are merged. The following ID values are potentially subject to collision:

7393

7394         *Certificates*  
7395         *SecurityDetails*  
7396         *SimplePart*  
7397         *Packaging*  
7398         *DocExchange*  
7399         *Transport*  
7400         *DeliveryChannel*  
7401         *ThisPartyActionBinding*

7402

7403     There are elements and complex type definitions containing IDREFs. Also some elements have  
7404     attributes with IDREF values. These are:

7405

7406         *PartyInfo*  
7407         *ActionBinding.type*  
7408         *ThisPartyActionBinding*  
7409         *OtherPartyActionBinding*  
7410         *OverrideMSHActionBinding*  
7411         *ChannelId*  
7412         *DeliveryChannel*  
7413         *Constituent*  
7414         *CertificateRef.type*  
7415         *AnchorCertificateRef*  
7416         *ApplicationCertificateRef*  
7417         *ClientCertificateRef*  
7418         *ServerCertificateRef*  
7419         *SigningCertificateRef*  
7420         *EncryptionCertificateRef*  
7421         *CertificateRef*  
7422         *SecurityDetailsRef.type*

7423

7424     Second, when the ***CanSend*** and ***CanReceive*** binding information has been found to match  
7425     (equal, correspond with, or be compatible with) the binding information under the other Party's

7426    **CanReceive** and **CanSend** elements, the IDREF references for the **OtherPartyActionBinding**  
7427    are filled out in the CPA.

7428  
7429    Third, for CPAs that are signed, the implementer is advised to review section 9.9.1.1 when using  
7430    [XMLDSIG] for the signature technique. A proposed CPA need not have a signature.  
7431

7432    Fourth, when a *CPA* is composed from two *CPPs*, see section 8.8 in which it stated that all  
7433    **Comment** elements from both *CPPs* SHALL be included in the *CPA* unless agreed to otherwise.  
7434

7435    Fifth, several tests on CPA validity could be conducted on draft CPAs, but these tests are more  
7436    critical for a negotiated CPA that is to be deployed and imported into run-time software  
7437    components.  
7438

7439    1. Expiration: Certificates used in signing a CPA can be checked to verify that they do not expire  
7440    before the CPA expires, as given in the **End** element.  
7441

7442    2. Certificate expiration: If a CPA lifetime exceeds the lifetime of certificates accepted for use in  
7443    signing, key exchange or other security functions, then it would be advisable to make ds:KeyInfo  
7444    refer to certificates, rather than to include them within the element by value.  
7445

7446    3. Process-Specification references can be checked in accordance with the provisions of section  
7447    8.4.4 and its subsections.  
7448

7449    Finally, a CPA has several elements whose values are not typically derived from either CPPs  
7450    (and can need checking when using a CPA template as the basis for a draft CPA.) The **Status**,  
7451    **Start**, **End**, and possibly a **ConversationConstraints** element need to be added. The attributes,  
7452

7453        **CollaborationProtocolAgreement/@cpaid**,  
7454        **CollaborationProtocolAgreement/@version**,  
7455        **CollaborationProtocolAgreement/Status@value**,  
7456        **CollaborationProtocolAgreement/ConversationConstrain@invocationLimit**, and  
7457        **CollaborationProtocolAgreement/ConversationConstraint@concurrentConversations**,  
7458

7459    can also be supplied values as needed.

## 7460 Appendix F Correspondence Between CPA and ebXML

### 7461 Messaging Parameters (Normative)

7462 The following table shows the correspondence between elements used in the ebXML Messaging  
 7463 Service message header and their counterparts in the CPA.  
 7464

Message Header Element / Attribute	Corresponding CPA Element / Attribute
<i>PartyId</i> element	<i>PartyId</i> element; if multiple <i>PartyID</i> elements occur under the same <i>PartyInfo</i> element in the CPA, all of them MUST be included in the <i>Message Header</i>
<i>Role</i> element	<i>Role</i> element
<i>CPAId</i> element	<i>cpaid</i> attribute in <i>CollaborationProtocolAgreement</i> element
<i>ConversationId</i> element	No equivalent; SHOULD be generated by software above the Message Service Interface (MSI)
<i>Service</i> element	<i>Service</i> element
<i>Action</i> element	<i>action</i> attribute in <i>ThisPartyActionBinding</i> element
<i>TimeToLive</i> element	Computed as the sum of <i>Timestamp</i> (in message header) + <i>PersistDuration</i> (under <i>DocExchange/ebXMLReceiverBinding</i> )
<i>MessageId</i> element	No equivalent; generated by the MSH per message
<i>Timestamp</i> element	No equivalent; generated by the MSH per message
<i>RefToMessageId</i> element	No equivalent; usually passed in by the application where applicable; SHOULD be used for correlating response messages with request messages
<i>SyncReply</i> element	<i>syncReplyMode</i> attribute in <i>MessagingCharacteristics</i> element; the <i>SyncReply</i> element is included if and only if the <i>syncReplyMode</i> attribute is not “none”
<i>DuplicateElimination</i> element	<i>duplicateElimination</i> attribute in <i>MessagingCharacteristics</i> element; the <i>DuplicateElimination</i> element is included if the <i>duplicateElimination</i> attribute under <i>MessagingCharacteristics</i> is set to “always”, or if it is set to “perMessage” and the application indicates to the MSH that duplicate elimination is desired
<i>Manifest</i> element	<i>Packaging</i> element; each <i>Reference</i> element under

	<b>Manifest</b> SHOULD correspond to a <b>SimplePart</b> that is referenced from one of the <b>CompositeList</b> elements under <b>Packaging</b>
<b>xlink:role</b> attribute in <b>Reference</b> element	<b>xlink:role</b> attribute in <b>SimplePart</b> element
<b>AckRequested</b> element	<b>ackRequested</b> attribute in <b>MessagingCharacteristics</b> element; an <b>AckRequested</b> element is included in the SOAP Header if the <b>ackRequested</b> attribute is set to “always”; if it is set to “perMessage”, input passed to the MSI is to be used to determine if an <b>AckRequested</b> element needs to be included; likewise, the signed attribute under <b>AckRequested</b> will be appropriately set based on the <b>ackSignatureRequested</b> attribute and possibly determined by input passed to the MSI
<b>MessageOrder</b> element	<b>messageOrderSemantics</b> attribute in <b>ReliableMessaging</b> element; the <b>MessageOrder</b> element will be present if the <b>AckRequested</b> element is present, and if the <b>messageOrderSemantics</b> attribute in the ReliableMessaging element is set to "Guaranteed"
<b>ds:Signature</b> element	<b>ds:Signature</b> will be present in the SOAP Header if the <b>isNonRepudiationRequired</b> attribute in the <b>BusinessTransactionCharacteristics</b> element is set to “true”; the relevant parameters for constructing the signature can be obtained from the <b>SenderNonRepudiation</b> and <b>ReceiverNonRepudiation</b> elements

7466  
7467  
7468  
7469  
7470

The following table shows the implicit parameters employed by the ebXML Messaging Service that are not included in the message header and how those parameters can be obtained from the CPA.

Implicit Messaging Parameters	Corresponding CPA Element / Attribute
<b>Retries</b> (not in Message Header) but used to govern Reliable Messaging behavior in sender	<b>Retries</b> element (under <b>ReliableMessaging</b> element)
<b>RetryInterval</b> (not in Message Header) but used to govern Reliable Messaging behavior in sender	<b>RetryInterval</b> element (under <b>ReliableMessaging</b> element)
<b>PersistDuration</b> (not in Message Header) but used to govern Reliable Messaging behavior in receiver	<b>PersistDuration</b> element (under <b>ebXMLReceiverBinding</b> element)
<b>Endpoint</b> (not in Message Header) but used for sending SOAP message	<b>Endpoint</b> element (under <b>TransportReceiver</b> ); the type of message

	being sent MUST be passed in to the MSI; an appropriate endpoint can then be selected from among the <i>Endpoints</i> included under the <i>TransportReceiver</i> element
Use <i>Service &amp; Action</i> to determine the corresponding <i>DeliveryChannel</i>	<i>DeliveryChannel</i>
Use <i>ReceiverDigitalEnvelope</i> to determine the encryption algorithm and key	<i>ReceiverDigitalEnvelope</i>
Use <i>SenderNonRepudiation</i> to determine signing certificate(s) and <i>ReceiverNonRepudiation</i> to determine the trust anchors and security policy to apply to the signing certificate	<i>SenderNonRepudiation</i> and <i>ReceiverNonRepudiation</i>
Use <i>Packaging</i> to determine how payload containers ought to be encapsulated. Also use <i>Packaging</i> to determine how an individual SimplePart ought to be extracted and validated against its schema	<i>Packaging</i>
Use <i>TransportClientSecurity</i> and <i>TransportServerSecurity</i> to determine certificates to be used by server and client for authentication purposes	<i>TransportClientSecurity</i> and <i>TransportServerSecurity</i>
Use the <i>DeliveryChannel</i> identified by <i>defaultMshChannelId</i> for standalone MSH level messages like Acknowledgment, Error, StatusRequest, StatusResponse, Ping, Pong, unless overridden by <i>OverrideMshActionBinding</i>	<i>defaultMshChannelId</i> attribute in <i>PartyInfo</i> element, and <i>OverrideMshActionBinding</i>

7471 **Appendix G Glossary of Terms**

7472

<b>Term</b>	<b>Definition</b>
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 <i>Server</i> .
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 state.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, organisation 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 <i>Message</i> .
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 <i>Business Transaction</i> .
RESPONDER	A counterpart to the initiator in a <i>Business Transaction</i> .
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 <i>Message</i> .
SERVER	Software that accepts a connection initiated by a <i>Client</i> .
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.
---	---

7473