



Creating A Single Global Electronic Market

1  
2  
3  
4 Collaboration-Protocol Profile and Agreement  
5 Specification  
6 Version 1.11

7  
8 OASIS ebXML Collaboration Protocol Profile  
9 and Agreement Technical Committee

10  
11 April 4 2002  
12  
13

14 1 Status of this Document

15  
16 This document specifies an ebXML SPECIFICATION for the eBusiness community.  
17

18 Distribution of this document is unlimited.  
19

20 The document formatting is based on the Internet Society's Standard RFC format.  
21

22 ***This version:***

23  
24 [http://www.oasis-open.org/committees/ebxml-cppa/documents/working\\_drafts/ebCPP-1\\_11.pdf](http://www.oasis-open.org/committees/ebxml-cppa/documents/working_drafts/ebCPP-1_11.pdf)  
25

26 ***Errata for this version:***

27  
28 [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

30 ***Previous version:***

31  
32 <http://www.ebxml.org/specs/ebCCP.pdf>  
33

34 **2 Technical Committee Members**

35	Selem Aissi	Intel
36	Arvola Chan	TIBCO
37	James Bryce Clark	McLure Moynihan
38	David Fischer	Drummond Group
39	Tony Fletcher	Individual Member
40	Brian Hayes	Commerce One
41	Neelakantan Kartha	Sterling Commerce
42	Kevin Liu	SAP
43	Pallavi Malu	Intel
44	Dale Moberg	Cyclone Commerce
45	Himagiri Mukkamala	Sybase
46	Peter Ogden	Cyclone Commerce
47	Marty Sachs	IBM
48	Yukinori Saito	Individual Member
49	David Smiley	Mercator
50	Tony Weida	Individual Member
51	Pete Wenzel	SeeBeyond
52	Jean Zheng	Vitria
53		

### 54 3 ebXML Participants

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

57

58 David Burdett, CommerceOne

59 Tim Chiou, United World Chinese Commercial Bank

60 Chris Ferris, Sun

61 Scott Hinkelman, IBM

62 Maryann Hondo, IBM

63 Sam Hunting, ECOM XML

64 John Ibbotson, IBM

65 Kenji Itoh, JASTPRO

66 Ravi Kacker, eXcelon Corp.

67 Thomas Limanek, iPlanet

68 Daniel Ling, VCHEQ

69 Henry Lowe, OMG

70 Dale Moberg, Cyclone Commerce

71 Duane Nickull, XMLGlobal Technologies

72 Stefano Pogliani, Sun

73 Rebecca Reed, Mercator

74 Karsten Riemer, Sun

75 Marty Sachs, IBM

76 Yukinori Saito, ECOM

77 Tony Weida, Edifecs

78

79	<b>4</b>	<b>Table of Contents</b>	
80	1	Status of this Document.....	1
81	2	Technical Committee Members .....	2
82	3	ebXML Participants .....	3
83	4	Table of Contents.....	4
84	5	Introduction.....	8
85	5.1	Summary of Contents of Document .....	8
86	5.2	Document Conventions.....	8
87	5.3	Versioning of the Specification and Schema .....	9
88	5.4	Definitions.....	10
89	5.5	Audience.....	10
90	5.6	Assumptions .....	10
91	5.7	Related Documents .....	10
92	6	Design Objectives .....	11
93	7	System Overview .....	12
94	7.1	What This Specification Does .....	12
95	7.2	Forming a CPA from Two CPPs.....	14
96	7.3	Forming a CPA from a CPA Template .....	16
97	7.4	How the CPA Works .....	16
98	7.5	Where the CPA May Be Implemented.....	17
99	7.6	Definition and Scope.....	18
100	8	CPP Definition .....	19
101	8.1	Globally-Unique Identifier of CPP Instance Document .....	19
102	8.2	CPP Structure .....	20
103	8.3	CollaborationProtocolProfile element.....	20
104	8.4	PartyInfo Element.....	21
105	8.4.1	PartyId element .....	23
106	8.4.2	PartyRef element.....	24
107	8.4.2.1	xlink:type attribute.....	24
108	8.4.2.2	xlink:href attribute.....	24
109	8.4.2.3	type attribute .....	24
110	8.4.2.4	schemaLocation attribute.....	24
111	8.4.3	CollaborationRole element.....	25
112	8.4.4	ProcessSpecification element .....	28
113	8.4.4.1	name attribute .....	29
114	8.4.4.2	version attribute .....	29
115	8.4.4.3	xlink:type attribute .....	29
116	8.4.4.4	xlink:href attribute.....	29
117	8.4.4.5	uuid attribute .....	29
118	8.4.4.6	ds:Reference element.....	29
119	8.4.5	Role element .....	31
120	8.4.5.1	name attribute .....	31
121	8.4.5.2	xlink:type attribute.....	32
122	8.4.5.3	xlink:href attribute.....	32
123	8.4.6	ApplicationCertificateRef element .....	32
124	8.4.6.1	certId attribute.....	32
125	8.4.7	ApplicationSecurityDetailsRef element.....	33
126	8.4.7.1	SecurityId attribute.....	33
127	8.4.8	ServiceBinding element.....	33
128	8.4.9	Service element .....	33
129	8.4.9.1	type attribute .....	33
130	8.4.10	CanSend element .....	34
131	8.4.11	CanReceive element.....	34

132	8.4.12 ThisPartyActionBinding element.....	34
133	8.4.12.1 action attribute.....	35
134	8.4.12.2 packageId attribute.....	36
135	8.4.12.3 xlink:href attribute.....	36
136	8.4.12.4 xlink:type attribute.....	36
137	8.4.13 BusinessTransactionCharacteristics element.....	36
138	8.4.13.1 isNonRepudiationRequired attribute.....	37
139	8.4.13.2 isNonRepudiationReceiptRequired attribute.....	37
140	8.4.13.3 isSecureTransportRequired attribute.....	37
141	8.4.13.4 isConfidential attribute.....	37
142	8.4.13.5 isAuthenticated attribute.....	38
143	8.4.13.6 isAuthorizationRequired attribute.....	38
144	8.4.13.7 isTamperProof attribute.....	38
145	8.4.13.8 isIntelligibleCheckRequired attribute.....	38
146	8.4.13.9 timeToAcknowledgeReceipt attribute.....	38
147	8.4.13.10 timeToAcknowledgeAcceptance attribute.....	39
148	8.4.13.11 timeToPerform attribute.....	39
149	8.4.13.12 retryCount attribute.....	39
150	8.4.14 ChannelId element.....	39
151	8.4.15 ActionContext element.....	39
152	8.4.15.1 binaryCollaboration attribute.....	40
153	8.4.15.2 businessTransactionActivity attribute.....	40
154	8.4.15.3 requestOrResponseAction attribute.....	40
155	8.4.16 CollaborationActivity element.....	41
156	8.4.16.1 name attribute.....	41
157	8.4.17 Certificate element.....	41
158	8.4.17.1 certId attribute.....	41
159	8.4.17.2 ds:KeyInfo element.....	42
160	8.4.18 SecurityDetails element.....	42
161	8.4.18.1 securityId attribute.....	42
162	8.4.19 TrustAnchors element.....	42
163	8.4.20 SecurityPolicy element.....	43
164	8.4.21 DeliveryChannel element.....	43
165	8.4.21.1 channelId attribute.....	45
166	8.4.21.2 transportId attribute.....	45
167	8.4.21.3 docExchangeId attribute.....	45
168	8.4.22 MessagingCharacteristics element.....	45
169	8.4.22.1 syncReplyMode attribute.....	45
170	8.4.22.2 ackRequested attribute.....	47
171	8.4.22.3 ackSignatureRequested attribute.....	47
172	8.4.22.4 duplicateElimination attribute.....	48
173	8.4.22.5 actor attribute.....	48
174	8.4.23 Transport element.....	49
175	8.4.23.1 transportId attribute.....	50
176	8.4.24 TransportSender element.....	50
177	8.4.25 TransportProtocol element.....	50
178	8.4.26 AccessAuthentication element.....	51
179	8.4.27 TransportClientSecurity element.....	51
180	8.4.28 TransportSecurityProtocol element.....	51
181	8.4.29 ClientCertificateRef element.....	52
182	8.4.30 ServerSecurityDetailsRef element.....	52
183	8.4.31 Encryption Algorithm.....	52
184	8.4.32 TransportReceiver element.....	53
185	8.4.33 Endpoint element.....	53
186	8.4.33.1 uri attribute.....	53
187	8.4.33.2 type attribute.....	53

188	8.4.34 TransportServerSecurity element.....	53
189	8.4.35 ServerCertificateRef element .....	54
190	8.4.36 ClientSecurityDetailsRef element.....	54
191	8.4.37 Transport protocols .....	54
192	8.4.37.1 HTTP.....	54
193	8.4.37.2 SMTP.....	55
194	8.4.37.3 FTP.....	55
195	8.4.38 DocExchange Element.....	56
196	8.4.38.1 docExchangeId attribute.....	57
197	8.4.39 ebXMLSenderBinding element .....	58
198	8.4.39.1 version attribute.....	58
199	8.4.40 ReliableMessaging element.....	58
200	8.4.40.1 Retries and RetryInterval elements.....	58
201	8.4.40.2 MessageOrderSemantics element .....	59
202	8.4.41 PersistDuration element.....	59
203	8.4.42 SenderNonRepudiation element.....	59
204	8.4.43 NonRepudiationProtocol element.....	60
205	8.4.44 HashFunction element .....	60
206	8.4.45 SignatureAlgorithm element.....	60
207	8.4.45.1 oid attribute.....	60
208	8.4.45.2 w3c attribute .....	60
209	8.4.45.3 enumeratedType attribute.....	61
210	8.4.46 SigningCertificateRef element .....	61
211	8.4.47 SenderDigitalEnvelope element.....	61
212	8.4.48 DigitalEnvelopeProtocol element .....	61
213	8.4.49 EncryptionAlgorithm element.....	61
214	8.4.49.1 minimumStrength attribute .....	62
215	8.4.49.2 oid attribute .....	62
216	8.4.49.3 w3c attribute .....	62
217	8.4.49.4 enumeratedTypeAttribute.....	62
218	8.4.50 EncryptionSecurityDetailsRef element.....	62
219	8.4.51 NamespaceSupported element .....	62
220	8.4.52 ebXMLReceiverBinding element .....	63
221	8.4.53 ReceiverNonRepudiation element .....	63
222	8.4.54 SigningSecurityDetailsRef element.....	64
223	8.4.55 ReceiverDigitalEnvelope element .....	64
224	8.4.56 EncryptionCertificateRef element .....	64
225	8.4.57 OverrideMshActionBinding element .....	64
226	8.5 SimplePart element.....	65
227	8.6 Packaging element.....	66
228	8.6.1 ProcessingCapabilities element.....	66
229	8.6.2 CompositeList element .....	67
230	8.7 Signature element .....	68
231	8.8 Comment element.....	69
232	9 CPA Definition.....	70
233	9.1 CPA Structure.....	70
234	9.2 CollaborationProtocolAgreement element .....	71
235	9.3 Status Element.....	71
236	9.4 CPA Lifetime .....	72
237	9.4.1 Start element .....	72
238	9.4.2 End element .....	72
239	9.5 ConversationConstraints Element.....	73
240	9.5.1 invocationLimit attribute.....	73
241	9.5.2 concurrentConversations attribute.....	74
242	9.6 PartyInfo Element.....	74
243	9.6.1 ProcessSpecification element .....	74

244	9.7 SimplePart element .....	74
245	9.8 Packaging element .....	74
246	9.9 Signature element .....	75
247	9.9.1 Persistent Digital Signature .....	75
248	9.9.1.1 Signature Generation .....	75
249	9.9.1.2 ds:SignedInfo element .....	76
250	9.9.1.3 ds:CanonicalizationMethod element .....	76
251	9.9.1.4 ds:SignatureMethod element .....	76
252	9.9.1.5 ds:Reference element .....	76
253	9.9.1.6 ds:Transform element .....	76
254	9.9.1.7 ds:Algorithm attribute .....	77
255	9.10 Comment element .....	77
256	9.11 Composing a CPA from Two CPPs .....	77
257	9.11.1 ID Attribute Duplication .....	77
258	9.12 Modifying Parameters of the Process-Specification Document Based on Information in the CPA .....	78
259	10 References .....	79
260	11 Conformance .....	82
261	12 Disclaimer .....	83
262	13 Contact Information .....	84
263	Notices .....	86
264	Copyright Statement .....	87
265	Appendix A Example of CPP Document (Non-Normative) .....	88
266	Appendix B Example of CPA Document (Non-Normative) .....	103
267	Appendix C Business Process Specification Corresponding to Complete CPP and CPA Definition (Non-Normative)	
268	.....	118
269	Appendix D W3C XML Schema Document Corresponding to Complete CPP and CPA Definition (Normative) ...	120
270	Appendix E CPA Composition (Non-Normative) .....	130
271	E.1 Suggestions for Design of Computational Procedures .....	130
272	E.2 CPA Formation Component Tasks .....	132
273	E.3 CPA Formation from <i>CPPs</i> : Context of Tasks .....	132
274	E.4 Business Collaboration Process Matching Tasks .....	133
275	E.5 Implementation Matching Tasks .....	134
276	E.6 CPA Formation: Technical Details .....	150
277	Appendix F Correspondence Between CPA and ebXML Messaging Parameters (Normative) .....	152
278	Appendix G Glossary of Terms .....	155
279		

## 280 5 Introduction

281

### 282 5.1 Summary of Contents of Document

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

292

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

295

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

297

298 The rest of this specification is organized as follows:

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

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

### 315 5.2 Document Conventions

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

318

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

321



322 References to external documents are represented with BLOCK text enclosed in brackets, e.g.  
323 [RFC2396]. The references are listed in Section 10, "References".  
324

325 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD  
326 NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be  
327 interpreted as described in [RFC 2119].  
328

329 NOTE: Vendors SHOULD carefully consider support of elements with cardinalities (0 or  
330 1) or (0 or more). Support of such an element means that the element is processed  
331 appropriately for its defined function and not just recognized and ignored. A given *Party*  
332 might use these elements in some *CPPs* or *CPAs* and not in others. Some of these elements  
333 define parameters or operating modes and SHOULD be implemented by all vendors. It  
334 might be appropriate to implement elective elements that represent major run-time  
335 functions, such as various alternative communication protocols or security functions, by  
336 means of plug-ins so that a given *Party* MAY acquire only the needed functions rather than  
337 having to install all of them.  
338

339 By convention, values of [XML] attributes are generally enclosed in quotation marks, however  
340 those quotation marks are not part of the values themselves.  
341

### 342 5.3 Versioning of the Specification and Schema

343 Whenever this specification is modified, it SHALL be given a new version number.  
344

345 It is anticipated that during the review period, errors and inconsistencies in the specification and  
346 in the schema may be detected and have to be corrected. All known errors in the specification as  
347 well as necessary changes to the schema will be summarized in an errata page found at  
348

349 [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)  
350

351 The specification, when initially approved by the OASIS ebXML Collaboration Protocol Profile  
352 and Agreement Technical Committee for public review, will carry a version number of "2\_0". At  
353 that time, the schema will have a version number of "2\_0a" and the suffix letter after "2\_0" will  
354 be advanced as necessary when bug fixes to the schema have to be introduced. Such versions of  
355 the schema will be found under the directory  
356

357 <http://www.oasis-open.org/committees/ebxml-cppa/schema/>  
358

359 In addition, the latest version of the schema can always be found at  
360

361 [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)  
362

363 since the latter is the namespace URI used for this specification and the corresponding schema is  
364 supposed to be directly resolvable from the namespace URI.  
365

366 The value of the version attribute of the Schema element in a given version of the schema  
367 SHALL be equal to the version of the schema.  
368

## 369 **5.4 Definitions**

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

371

## 372 **5.5 Audience**

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

377

## 378 **5.6 Assumptions**

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

381

## 382 **5.7 Related Documents**

383 Related documents include ebXML Specifications on the following topics:

384

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

389

390 See Section 10 for the complete list of references.

391

## 392 6 Design Objectives

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

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

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

414  
415 It is a further objective of this specification to facilitate migration of both traditional EDI-based  
416 applications and other legacy applications to platforms based on the ebXML specifications. In  
417 particular, the *CPP* and *CPA* are components of the migration of applications based on the X12  
418 838 Trading-Partner Profile[X12] to more automated means of setting up *Business* relationships  
419 and doing *Business* under them.

## 420 7 System Overview

### 421 7.1 What This Specification Does

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

426  
427 The way each *Party* can exchange information, in the context of a *Business Collaboration*, can  
428 be described by a *Collaboration-Protocol Profile (CPP)*. The agreement between the *Parties* can  
429 be expressed as a *Collaboration-Protocol Agreement (CPA)*.

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

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

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

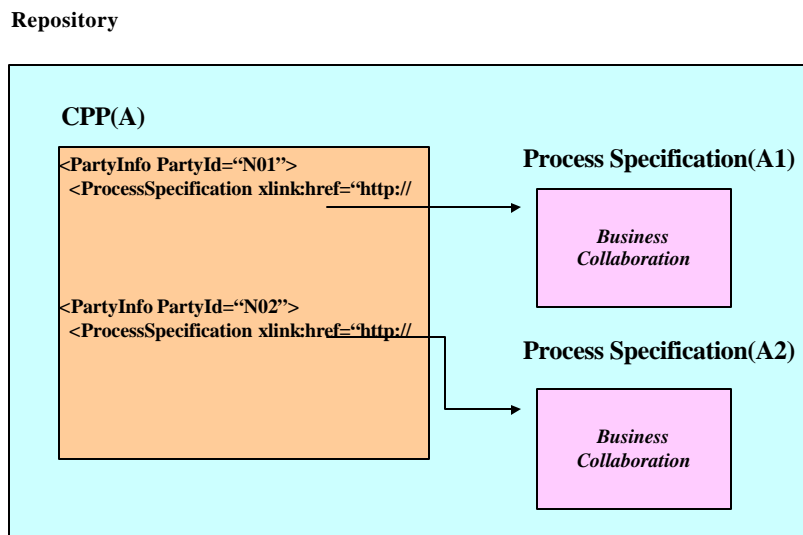
445  
446 Figure 1 illustrates the relationships between a *CPP* and two *Process-Specification* documents,  
447 A1 and A2, in an ebXML Registry. On the left is a *CPP*, A, which includes information about  
448 two parts of an enterprise that are represented as different *Parties*. On the right are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

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

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

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

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



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

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

- 480  
 481
- A *Client* initiates a connection with a *Server*.
  - A *Requester* initiates a *Business Transaction* with a *Responder*.
  - A *Sender* sends a *Message* to a *Receiver*.
- 482  
 483  
 484

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

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

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

503

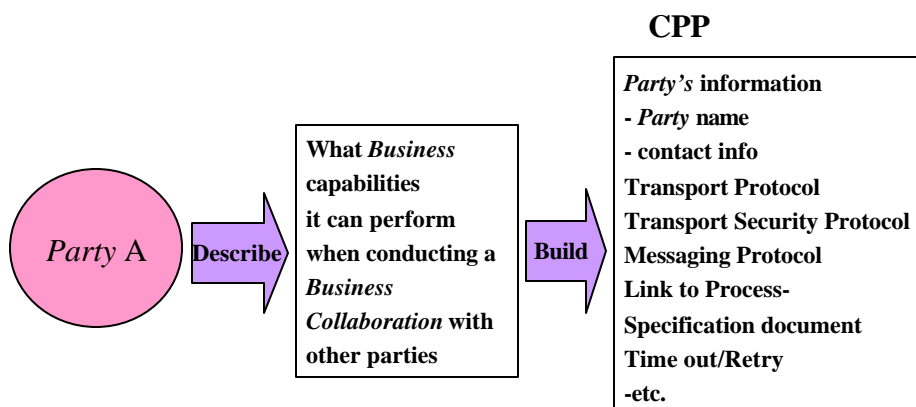
## 504 **7.2 Forming a CPA from Two CPPs**

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

509

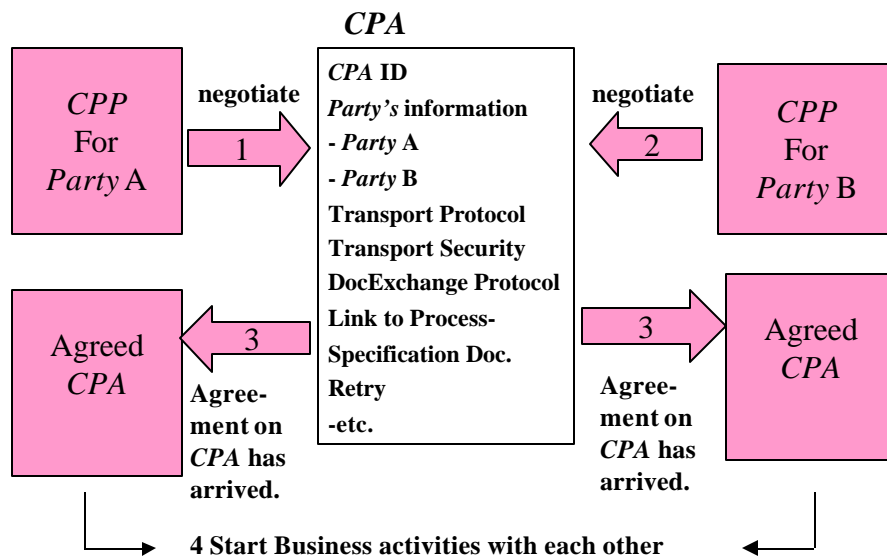
510 Figure 2 illustrates forming a CPP. *Party A* tabulates the information to be placed in a repository  
511 for the discovery process, constructs a CPP that contains this information, and enters it into an  
512 ebXML Registry or similar repository along with additional information about the *Party*.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

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



516  
 517 In Figure 3, *Party A* and *Party B* use their *CPPs* to jointly construct a single copy of a *CPA* by  
 518 calculating the intersection of the information in their *CPPs*. The resulting *CPA* defines how the  
 519 two *Parties* will behave in performing their *Business Collaboration*.

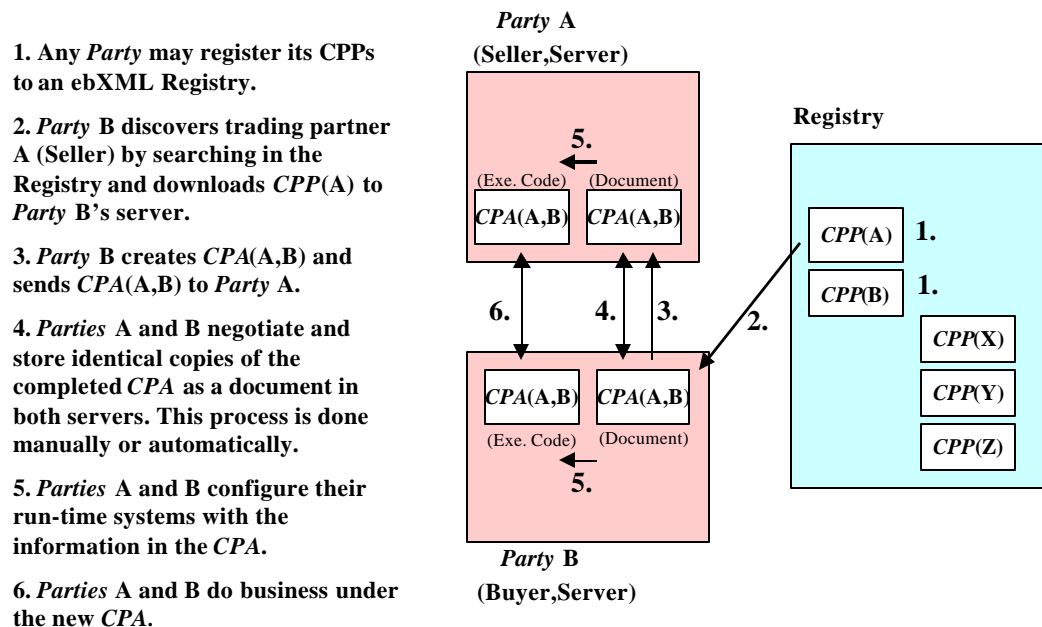
**Figure 3: Overview of Collaboration-Protocol Agreements (CPA)**



520

521 Figure 4 illustrates the entire process. The steps are listed at the left. The end of the process is  
 522 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



523 then ready to do *Business*.

524

### 525 7.3 Forming a CPA from a CPA Template

526 Alternatively, a *CPA* template might be used to create a *CPA*. A *CPA* template represents one  
 527 party's "fill in the blanks" proposal to a prospective trading partner for implementing one or  
 528 more business processes. For example, such a template might contain placeholder values for  
 529 identifying aspects of the other party. To form a *CPA* from a *CPA* template, the placeholder  
 530 values would be replaced by the actual values for the other trading partner. Actual values might  
 531 be obtained from the other party's *CPP*, if available, or by data entry in an HTML form, among  
 532 other possibilities. The current version of this specification does not address how placeholder  
 533 values might be represented in a *CPA*. However, the process of filling out a *CPA* template  
 534 MUST result in a valid *CPA*. Further discussion of *CPA* templates is provided in Appendix E.  
 535

### 536 7.4 How the CPA Works

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



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

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

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

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

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

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

575

## 576 **7.5 Where the CPA May Be Implemented**

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

586 other information in the *CPA* and *Process-Specification* document MAY be used in generating or  
587 customizing the necessary code to support the *CPA*.  
588

589 NOTE: It is possible to provide a graphical *CPP/CPA*-authoring tool that understands both  
590 the semantics of the *CPP/CPA* and the XML syntax. Equally important, the definitions in  
591 this specification make it feasible to automatically generate, at each *Party's* site, the code  
592 needed to execute the *CPA*, enforce its rules, and interface with the *Party's* back-end  
593 processes.  
594

## 595 **7.6 Definition and Scope**

596 This specification defines and explains the contents of the *CPP* and *CPA* XML documents. Its  
597 scope is limited to these definitions. It does not define how to compose a *CPA* from two *CPPs*  
598 nor does it define anything related to run-time support for the *CPP* and *CPA*. It does include  
599 some non-normative suggestions and recommendations regarding *CPA* composition from two  
600 *CPPs* and run-time support where these notes serve to clarify the *CPP* and *CPA* definitions. See  
601 Section 11 for a discussion of conformance to this specification.  
602

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

## 609 8 CPP Definition

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

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

618  
619 The ***ProcessSpecification***, ***DeliveryChannel***, ***DocExchange***, and ***Transport*** elements of the  
620 *CPP* describe the processing of a unit of *Business* (conversation). These elements form a layered  
621 structure somewhat analogous to a layered communication model.

622  
623 **Process-Specification layer** - The *Process-Specification* layer defines the heart of the *Business*  
624 agreement between the *Parties*: the services (*Business Transactions*) which *Parties* to the *CPA*  
625 can request of each other and transition rules that determine the order of requests. This layer is  
626 defined by the separate *Process-Specification* document that is referenced by the *CPP* and *CPA*.

627  
628 **Delivery Channels** - A delivery channel describes a *Party's* *Message*-receiving and *Message*-  
629 sending characteristics. It consists of one document-exchange definition and one transport  
630 definition. Several delivery channels MAY be defined in one *CPP*.

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

640  
641 **Transport layer** - The transport layer identifies the transport protocol to be used in sending  
642 messages through the network and defines the endpoint addresses, along with various other  
643 properties of the transport protocol. Choices of properties in the transport layer are  
644 complementary to those in the document-exchange layer (see "Document-Exchange Layer"  
645 directly above.)

646  
647 Note that the functional layers encompassed by the *CPP* are independent of the contents of the  
648 payload of the *Business* documents.

649

### 650 8.1 Globally-Unique Identifier of CPP Instance Document

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

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

654

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

658

## 659 8.2 CPP Structure

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

662

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

688

## 689 8.3 CollaborationProtocolProfile element

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

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

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

698

699 In addition, the *CollaborationProtocolProfile* element contains a REQUIRED *cppid* attribute  
700 that supplies a unique identifier for the document, plus a REQUIRED *version* attribute that

701 indicates the version of the schema. Its purpose is to identify the version of the schema that the  
 702 *CPP* conforms to. The value of the version attribute SHOULD be a string such as "2\_0a" or  
 703 "2\_0b". The value of the cppid attribute SHOULD be changed with each change made to the  
 704 *CPP* document after it has been published.

705  
 706 NOTE: The method of assigning unique cppid values is left to the implementation.

707  
 708 The ***CollaborationProtocolProfile*** element SHALL consist of the following child elements:

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

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

## 724 8.4 PartyInfo Element

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

```

730
731 <tp:PartyInfo
732     tp:partyName="..."
733     tp:defaultMshChannelId="..."
734     tp:defaultMshPackageId="...">
735     <tp:PartyId tp:type="..."> <!-- one or more -->
736     ...
737     </tp:PartyId>
738     <tp:PartyRef xlink:type="..." xlink:href="..." />
739     <tp:CollaborationRole> <!-- one or more -->
740     ...
741     </tp:CollaborationRole>
742     <tp:Certificate> <!-- one or more -->
743     ...
744     </tp:Certificate>
745     <tp:SecurityDetails> <!-- one or more -->
746     ...
747     </tp:SecurityDetails>
748     <tp:DeliveryChannel> <!-- one or more -->
749     ...
  
```

```

750         </tp:DeliveryChannel>
751         <tp:Transport>  <!-- one or more -->
752         ...
753         </tp:Transport>
754         <tp:DocExchange>  <!-- one or more -->
755         ...
756         </tp:DocExchange>
757         </tp:OverrideMshActionBinding>  <!-- zero or more -->
758         ...
759         </tp:OverrideMshActionBinding>
760     </tp:PartyInfo>
761

```

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

766

767 The following example illustrates two possible party names.

768

```

769         <tp:PartyInfo tp:partyName="Example, Inc."...</tp:PartyInfo>
770
771         <tp:PartyInfo tp:partyName="Example, Inc. US Western Division">
772         ...
773         </tp:PartyInfo>
774

```

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

784

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

- 786 • One or more REQUIRED **PartyId** elements that provide a logical identifier for the  
787 organization.
- 788 • One or more REQUIRED **PartyRef** element that provides a pointer to more  
789 information about the **Party**.
- 790 • One or more REQUIRED **CollaborationRole** elements that identify the roles that this  
791 **Party** can play in the context of a **Process Specification**.
- 792 • One or more REQUIRED **Certificate** elements that identify the certificates used by  
793 this **Party** in security functions.
- 794 • One or more REQUIRED **SecurityDetails** elements that identify trust anchors and  
795 specify security policy used by this **Party** in security functions.
- 796 • One or more REQUIRED **DeliveryChannel** elements that define the characteristics of  
797 each delivery channel that the **Party** can use to send and/or receive **Messages**. It  
798 includes both the transport protocol (e.g. HTTP) and the messaging protocol (e.g.

- 799 ebXML *Message Service*).
- 800 • One or more REQUIRED *Transport* elements that define the characteristics of the
  - 801 transport protocol(s) that the *Party* can support to send and receive *Messages*.
  - 802 • One or more REQUIRED *DocExchange* elements that define the *Message*-exchange
  - 803 characteristics, such as the *Message*-exchange protocol, that the *Party* can support.
  - 804 • Zero or more *OverrideMshActionBinding* elements that specify the *DeliveryChannel*
  - 805 to use for asynchronously delivered *Message Service Handler* level messages.
  - 806

#### 807 8.4.1 PartyId element

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

813

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

817

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

822

823 The *PartyId* element has a single IMPLIED attribute: *type* that has an anyURI [XMLSCHEMA-  
 824 2] value.

825

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

828

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

833

834 The following example illustrates two URI references.

835

```
836 <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-
837 type:duns">123456789</tp:PartyId>
838
839 <tp:PartyId>urn:icann:example.com</tp:PartyId>
```

840

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

843

844 The second example shows an arbitrary URN. This might be a URN that the *Party* has

845 registered with IANA to identify itself directly.

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

849  
850 [http://www.oasis-open.org/committees/ebxml-cppa/documents/PartyID\\_Types.shtml](http://www.oasis-open.org/committees/ebxml-cppa/documents/PartyID_Types.shtml)  
851

## 852 **8.4.2 PartyRef element**

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

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

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

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

873

### 874 **8.4.2.1 xlink:type attribute**

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

877

### 878 **8.4.2.2 xlink:href attribute**

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

881

### 882 **8.4.2.3 type attribute**

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

887

### 888 **8.4.2.4 schemaLocation attribute**

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



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

```
893
894     <tp:PartyRef xlink:type="simple"
895               xlink:href="http://example2.com/ourInfo.xml"
896               tp:type="urn:oasis:names:tc:ebxml-cppa:contact-info"
897               tp:schemaLocation="http://example2.com/ourInfo.xsd"/>
898
```

### 899 8.4.3 CollaborationRole element

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

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

```

```

944         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
945     </tp:ThisPartyActionBinding>
946 </tp:CanSend>
947 <tp:CanSend>
948     <tp:ThisPartyActionBinding
949         tp:id="companyA_ABID2"
950         tp:action="ReceiptAcknowledgment"
951         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
952     <tp:BusinessTransactionCharacteristics
953         tp:isNonRepudiationRequired="true"
954         tp:isNonRepudiationReceiptRequired="true"
955         tp:isSecureTransportRequired="true"
956         tp:isConfidential="transient"
957         tp:isAuthenticated="persistent"
958         tp:isTamperProof="persistent"
959         tp:isAuthorizationRequired="true"
960         tp:timeToAcknowledgeReceipt="PT2H"
961         tp:timeToPerform="P1D"/>
962     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
963 </tp:ThisPartyActionBinding>
964 </tp:CanSend>
965 <tp:CanReceive>
966     <tp:ThisPartyActionBinding
967         tp:id="companyA_ABID3"
968         tp:action="Purchase Order Confirmation Action"
969         tp:packageId="CompanyA_ResponsePackage">
970     <tp:BusinessTransactionCharacteristics
971         tp:isNonRepudiationRequired="true"
972         tp:isNonRepudiationReceiptRequired="true"
973         tp:isSecureTransportRequired="true"
974         tp:isConfidential="transient"
975         tp:isAuthenticated="persistent"
976         tp:isTamperProof="persistent"
977         tp:isAuthorizationRequired="true"
978         tp:timeToAcknowledgeReceipt="PT2H"
979         tp:timeToPerform="P1D"/>
980     <tp:ActionContext
981         tp:binaryCollaboration="Request Purchase Order"
982         tp:businessTransactionActivity="Request Purchase
983 Order"
984         tp:requestOrResponseAction="Purchase Order
985 Confirmation Action"/>
986     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
987 </tp:ThisPartyActionBinding>
988 </tp:CanReceive>
989 <tp:CanReceive>
990     <tp:ThisPartyActionBinding
991         tp:id="companyA_ABID4"
992         tp:action="ReceiptAcknowledgment"
993         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
994     <tp:BusinessTransactionCharacteristics
995         tp:isNonRepudiationRequired="true"
996         tp:isNonRepudiationReceiptRequired="true"
997         tp:isSecureTransportRequired="true"
998         tp:isConfidential="transient"
999         tp:isAuthenticated="persistent"
1000        tp:isTamperProof="persistent"

```

```

1001         tp:isAuthorizationRequired="true"
1002         tp:timeToAcknowledgeReceipt="PT2H"
1003         tp:timeToPerform="P1D" />
1004     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
1005 </tp:ThisPartyActionBinding>
1006 </tp:CanReceive>
1007 <tp:CanReceive>
1008     <tp:ThisPartyActionBinding
1009         tp:id="companyA_ABID5"
1010         tp:action="Exception"
1011         tp:packageId="CompanyA_ExceptionPackage">
1012     <tp:BusinessTransactionCharacteristics
1013         tp:isNonRepudiationRequired="true"
1014         tp:isNonRepudiationReceiptRequired="true"
1015         tp:isSecureTransportRequired="true"
1016         tp:isConfidential="transient"
1017         tp:isAuthenticated="persistent"
1018         tp:isTamperProof="persistent"
1019         tp:isAuthorizationRequired="true"
1020         tp:timeToAcknowledgeReceipt="PT2H"
1021         tp:timeToPerform="P1D" />
1022     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
1023 </tp:ThisPartyActionBinding>
1024 </tp:CanReceive>
1025 </tp:ServiceBinding>
1026 </tp:CollaborationRole>
1027

```

1028 To indicate that the *Party* can play roles in more than one *Business Collaboration* or more than  
1029 one role in a given *Business Collaboration*, the ***PartyInfo*** element SHALL contain more than  
1030 one ***CollaborationRole*** element. Each ***CollaborationRole*** element SHALL contain the  
1031 appropriate combination of ***ProcessSpecification*** element and ***Role*** element.

1032  
1033 The ***CollaborationRole*** element SHALL consist of the following child elements: a REQUIRED  
1034 ***ProcessSpecification*** element, a REQUIRED ***Role*** element, zero or more  
1035 ***ApplicationCertificateRef*** elements, zero or one ***ApplicationSecurityDetailsRef*** element, and  
1036 one ***ServiceBinding*** element. The ***ProcessSpecification*** element identifies the *Process-*  
1037 *Specification* document that defines such role. The ***Role*** element identifies which role the *Party*  
1038 is capable of supporting. The ***ApplicationCertificateRef*** element identifies the certificate to be  
1039 used for application level signature and encryption. The ***ApplicationSecurityDetailsRef*** element  
1040 identifies the trust anchors and security policy that will be applied to any application-level  
1041 certificate offered by the other *Party*. The ***ServiceBinding*** element SHALL consist of zero or  
1042 more ***CanSend*** elements and zero or more ***CanReceive*** elements. The ***CanSend*** and ***CanReceive***  
1043 elements identify the ***DeliveryChannel*** elements that are to be used for sending and receiving  
1044 business action messages by the ***Role*** in question. They MAY also be used for specifying  
1045 ***DeliveryChannels*** for business signal messages.

1046  
1047 Each *Party* SHALL have a default delivery channel for the delivery of standalone *Message*  
1048 Service Handler level signals like (Reliable Messaging) Acknowledgments, Errors,  
1049 StatusRequest, StatusResponse, etc.

1050

1051 **8.4.4 ProcessSpecification element**

1052 The **ProcessSpecification** element provides the link to the *Process-Specification* document that  
 1053 defines the interactions between the two *Parties*. It is RECOMMENDED that this *Business-*  
 1054 *Collaboration* description be prepared in accordance with the ebXML Business Process  
 1055 Specification Schema[ebBPSS]. The *Process-Specification* document MAY be kept in an  
 1056 ebXML Registry.

1057  
 1058 NOTE: A *Party* can describe the *Business Collaboration* using any desired alternative to  
 1059 the ebXML Business Process Specification Schema. When an alternative *Business-*  
 1060 *Collaboration* description is used, the *Parties* to a *CPA* MUST agree on how to interpret  
 1061 the *Business-Collaboration* description and how to interpret the elements in the *CPA* that  
 1062 reference information in the *Business-Collaboration* description. The affected elements  
 1063 in the *CPA* are the **Role** element, the **CanSend** and **CanReceive** elements, the  
 1064 **ActionContext** element, and some attributes of the **BusinessTransactionCharacteristics**  
 1065 element.

1066  
 1067 The syntax of the **ProcessSpecification** element is:

```

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

1085  
 1086 The **ProcessSpecification** element has zero or more child **ds:Reference** elements, and the  
 1087 following attributes:

- 1088 • a REQUIRED **name** attribute,
- 1089 • a REQUIRED **version** attribute,
- 1090 • a FIXED **xlink:type** attribute,
- 1091 • a REQUIRED **xlink:href** attribute,
- 1092 • an IMPLIED **uuid** attribute.

1093  
 1094 The **ProcessSpecification** element contains zero or more **ds:Reference** elements formulated  
 1095 according to the XML Digital Signature specification[XMLDSIG]. The first **ds:Reference**  
 1096 element, if present, relates to the **xlink:type** and **xlink:href** attributes as follows. Each  
 1097 **ProcessSpecification** element SHALL contain one **xlink:href** attribute and one **xlink:type**  
 1098 attribute with a value of "simple". In case the *CPP* (*CPA*) document is signed, the first

1099 **ds:Reference** element that is present MUST include a **ds:URI** attribute whose value is identical  
1100 to that of the **xlink:href** attribute in the enclosing **ProcessSpecification** element. The  
1101 **ds:Reference** element specifies a digest method and digest value to enable verification that the  
1102 referenced *Process-Specification* document has not changed. Additional **ds:Reference** elements  
1103 are needed if the referenced **ProcessSpecification** in turn includes (i.e., references) other  
1104 **ProcessSpecifications**. Essentially, **ds:Reference** elements MUST be provided to correspond to  
1105 the transitive closure of all **ProcessSpecifications** that are referenced directly or indirectly to  
1106 ensure that none of them has been changed.

1107

#### 1108 **8.4.4.1 name attribute**

1109 The **ProcessSpecification** element MUST include a REQUIRED **name** attribute: a string that  
1110 identifies the *Business Process-Specification* being performed. If the *Process-Specification*  
1111 document is defined by the ebXML Business Process specification [ebBPSS], then this attribute  
1112 MUST be set to the **name** for the corresponding **ProcessSpecification** element within the  
1113 *Business Process Specification* instance.

1114

#### 1115 **8.4.4.2 version attribute**

1116 The **ProcessSpecification** element includes a REQUIRED **version** attribute to identify the  
1117 version of the *Process-Specification* document identified by the **xlink:href** attribute (and also  
1118 identified by the **ds:Reference** element, if any).

1119

#### 1120 **8.4.4.3 xlink:type attribute**

1121 The **xlink:type** attribute has a FIXED value of "simple". This identifies the element as being an  
1122 [XLINK] simple link.

1123

#### 1124 **8.4.4.4 xlink:href attribute**

1125 The REQUIRED **xlink:href** attribute SHALL have a value that identifies the *Process-*  
1126 *Specification* document and is a URI that conforms to [RFC2396].

1127

#### 1128 **8.4.4.5 uuid attribute**

1129 The IMPLIED **uuid** attribute uniquely identifies the **ProcessSpecification**. If the *Process-*  
1130 *Specification* document is defined by the ebXML Business Process specification [ebBPSS], then  
1131 this attribute MUST be set to the **uuid** for the corresponding **ProcessSpecification** element  
1132 within the business process specification instance.

1133

#### 1134 **8.4.4.6 ds:Reference element**

1135 The **ds:Reference** element identifies the same *Process-Specification* document as the enclosing  
1136 **ProcessSpecification** element's **xlink:href** attribute and additionally provides for verification that  
1137 the *Process-Specification* document has not changed since the *CPP* was created, through the use  
1138 of a digest method and digest value as described below.

1139

1140 NOTE: *Parties* MAY test the validity of the *CPP* or *CPA* at any time. The following  
1141 validity tests MAY be of particular interest:

1142

- 1143 • test of the validity of a *CPP* and the referenced *Process-Specification* documents at  
1144 the time composition of a *CPA* begins in case they have changed since they were

- 1145 created,
- 1146 • test of the validity of a *CPA* and the referenced *Process-Specification* documents at
- 1147 the time a *CPA* is installed into a *Party's* system,
- 1148 • test of the validity of a *CPA* at intervals after the *CPA* has been installed into a *Party's*
- 1149 system. The *CPA* and the referenced *Process-Specification* documents MAY be
- 1150 processed by an installation tool into a form suited to the particular middleware.
- 1151 Therefore, alterations to the *CPA* and the referenced *Process-Specification* documents
- 1152 do not necessarily affect ongoing run-time operations. Such alterations might not be
- 1153 detected until it becomes necessary to reinstall the *CPA* and the referenced *Process-*
- 1154 *Specification* documents.

1155

1156 The syntax and semantics of the *ds:Reference* element and its child elements are defined in the

1157 XML Digital Signature specification[XMLDSIG]. In addition, to identify the *Process-*

1158 *Specification* document, the *ds:Reference* MUST include a *ds:URI* attribute whose value is

1159 identical to that of the *xlink:href* attribute in the enclosing *ProcessSpecification* element.

1160

1161 According to [XMLDSIG], a *ds:Reference* element can have a *ds:Transforms* child element,

1162 which in turn has an ordered list of one or more *ds:Transform* child elements to specify a

1163 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 *ds:Signature* element.

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" (emphases 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* subelement 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

##### 1230 8.4.5.1 name attribute

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

- 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

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

1243

#### 1244 8.4.5.2 *xlink:type* attribute

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

1247

#### 1248 8.4.5.3 *xlink:href* attribute

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

1252

```
1253     xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"
```

1254

1255 Where "Buyer" is the value of the ID attribute of the element in the *Process-Specification*  
1256 document that defines the role name.

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

- 1272 • A REQUIRED *certId* attribute.

1273

#### 1274 8.4.6.1 *certId* attribute

1275 The REQUIRED *certId* attribute is an [XML] IDREF that associates the *CollaborationRole* with  
1276 a certificate. It MUST have a value equal the value of the *certId* attribute of one of the  
1277 *Certificate* elements under *PartyInfo*.

1278



#### 1279 **8.4.7 ApplicationSecurityDetailsRef element**

1280 The *ApplicationSecurityDetailsRef* element, if present, identifies the trust anchors and security  
1281 policy that this *Party* will apply to any application-level certificate offered by the other *Party*.  
1282 These trust anchors and policy are not used by the ebXML messaging system, but are included in  
1283 the *CPP* so that they can be considered in the *CPA* negotiation process.

1284  
1285 The *ApplicationSecurityDetailsRef* element has

- 1286 • A REQUIRED *securityId* attribute.

1287

##### 1288 **8.4.7.1 SecurityId attribute**

1289 The REQUIRED *securityId* attribute is an [XML] IDREF that associates the *CollaborationRole*  
1290 with a *SecurityDetails* element that specifies a set of trust anchors and a security policy. It  
1291 MUST have a value equal to the value of the *securityId* attribute of one of the *SecurityDetails*  
1292 elements under *PartyInfo*.

1293

#### 1294 **8.4.8 ServiceBinding element**

1295 The *ServiceBinding* element identifies a *DeliveryChannel* element for all of the business  
1296 *Message* traffic that is to be sent or received by the *Party* within the context of the identified  
1297 *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) specified for  
1310 the *ProcessSpecification* Element in the Business Process Specification Schema instance  
1311 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 *BusinessTransactionCharacteristics*  
1366 element, zero or one child *ActionContext* element and one or more *ChannelID* child elements.

1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412

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

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

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

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

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

#### 8.4.12.1 action attribute

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

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

NOTE: In general, the action name chosen by the two parties to represent a particular requesting business activity or responding business activity in the context of a binary collaboration that makes use of nested binary collaborations MAY not be identical.

Therefore, when composing two ***CPPs*** to form a ***CPA***, it is necessary to make use of information from the associated ***ActionContext*** (see Section 8.4.15) in order to determine

1413 if two different action names from the two *CPPs* actually represent the same  
1414 *ActionContext*. When business transactions are not reused in different contexts, it is  
1415 recommended that the names of the requesting business activity and responding business  
1416 activity be used as action names.

1417

#### 1418 **8.4.12.2 packageId attribute**

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

1421

#### 1422 **8.4.12.3 xlink:href attribute**

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

1427

#### 1428 **8.4.12.4 xlink:type attribute**

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

1431

### 1432 **8.4.13 BusinessTransactionCharacteristics element**

1433 The *BusinessTransactionCharacteristics* element describes the security characteristics and other  
1434 attributes of the delivery channel, as derived from the *ProcessSpecification(s)* whose messages  
1435 are transported using the delivery channel. The attributes of the  
1436 *BusinessTransactionCharacteristics* element, MAY be used to override the values of the  
1437 corresponding attributes in the *Process-Specification* document.

1438

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

1440

1441 *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel  
1442 definitions for the sender and receiver (transport and document-exchange) for internal  
1443 consistency as well as compatibility between the two partners. Typically, when an attribute has a  
1444 particular value, sub-elements under the corresponding Transport and DocExchange elements  
1445 would exist to further describe the implied implementation parameters.

1446

1447 The *BusinessTransactionCharacteristics* element has the following attributes:

1448

- 1449 • an IMPLIED *isNonRepudiationRequired* attribute,
- 1450 • an IMPLIED *isNonRepudiationReceiptRequired* attribute,
- 1451 • an IMPLIED *isSecureTransportRequired* attribute,
- 1452 • an IMPLIED *isConfidential* attribute,
- 1453 • an IMPLIED *isAuthenticated* attribute,
- 1454 • an IMPLIED *isAuthorizationRequired* attribute,
- 1455 • an IMPLIED *isTamperProof* attribute,
- 1456 • an IMPLIED *isIntelligibleCheckRequired* attribute,
- 1457 • an IMPLIED *timeToAcknowledgeReceipt* attribute,

- 1458           • an IMPLIED *timeToAcknowledgeAcceptance* attribute,  
1459           • an IMPLIED *timeToPerform* attribute,  
1460           • an IMPLIED *retryCount* attribute.

1461  
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*.

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 **8.4.13.2 isNonRepudiationReceiptRequired attribute**

1477 The *isNonRepudiationReceiptRequired* attribute is a Boolean with possible values of "true"  
1478 and "false". If the value is "true" then the delivery channel MUST specify that the *Message* is to  
1479 be acknowledged by a digitally signed *Receipt Acknowledgment* signal *Message*, signed using  
1480 the certificate of the *Party* that received the *Message*, that includes the digest(s) of the *Message*  
1481 being acknowledged. The *SenderNonRepudiation* element under  
1482 *DocExchange/ebXMLSenderBinding* (see Section 8.4.42) and the *ReceiverNonRepudiation*  
1483 element under *DocExchange/ebXMLReceiverBinding* (see Section 8.4.53) further describe  
1484 various parameters related to the implementation of non-repudiation of receipt.

1485

#### 1486 **8.4.13.3 isSecureTransportRequired attribute**

1487 The *isSecureTransportRequired* attribute is a Boolean with possible values of "true" and  
1488 "false". If the value is "true" then it indicates that the delivery channel uses a secure transport  
1489 protocol such as [SSL] or [IPSEC].

1490

#### 1491 **8.4.13.4 isConfidential attribute**

1492 The *isConfidential* attribute has the possible values of "none", "transient", "persistent", and  
1493 "transient-and-persistent". These values MUST be interpreted as defined by the ebXML Business  
1494 Process Specification Schema[ebBPSS]. In general, transient confidentiality can be implemented  
1495 using a secure transport protocol like SSL; persistent confidentiality can be implemented using a  
1496 digital envelope mechanism like S/MIME. Secure transport information is further provided in the  
1497 *TransportSender* (see Section 8.4.24) and *TransportReceiver* (see Section 8.4.31) elements  
1498 under the *Transport* element. Persistent encryption information is further provided in the  
1499 *SenderDigitalEnvelope* element under *DocExchange/ebXMLSenderBinding* (see Section  
1500 8.4.47) and the *ReceiverDigitalEnvelope* element under *DocExchange/ebXMLReceiverBinding*  
1501 (see Section 8.4.55).

1502

1503 The *CPA* would be inconsistent if *isConfidential* is set to "transient" or "persistent-and-

1504 transient", while *isSecureTransportRequired* is set to "false".

1505

#### 1506 **8.4.13.5 isAuthenticated attribute**

1507 The *isAuthenticated* attribute has the possible values of "none", "transient", "persistent", and  
1508 "persistent-and-transient". If this attribute is set to any value other than "none", then the receiver  
1509 MUST be able to verify the identity of the sender. In general, transient authentication can be  
1510 implemented using a secure transport protocol like SSL (with or without the use of basic or  
1511 digest authentication); persistent authentication can be implemented using a digital signature  
1512 mechanism. Secure transport information is further provided in the *TransportSender* (see  
1513 Section 8.4.24) and *TransportReceiver* (see Section 8.4.32) elements under the *Transport*  
1514 element. Persistent authentication information is further provided in the *SenderNonRepudiation*  
1515 element under *DocExchange/ebXMLSenderBinding* (see Section 8.4.42) and the  
1516 *ReceiverNonRepudiation* element (under *DocExchange/ebXMLReceiverBinding* (see Section  
1517 8.4.53).

1518

1519 The CPA would be inconsistent if *isAuthenticated* is set to "transient" or "persistent-and-  
1520 transient", while *isSecureTransportRequired* is set to "false".

1521

#### 1522 **8.4.13.6 isAuthorizationRequired attribute**

1523 The *isAuthorizationRequired* attribute is a Boolean with possible values of "true" and  
1524 "false". If the value is "true" then it indicates that the delivery channel MUST specify that the  
1525 sender of the *Message* is to be authorized before delivery to the application.

1526

#### 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

#### 1542 **8.4.13.8 isIntelligibleCheckRequired attribute**

1543 The *isIntelligibleCheckRequired* attribute is a Boolean with possible values of "true" and  
1544 "false". If the value is "true", then the receiver MUST verify that a business document is not  
1545 garbled (i.e., passes schema validation) before returning a *Receipt Acknowledgment* signal.

1546

#### 1547 **8.4.13.9 timeToAcknowledgeReceipt attribute**

1548 The *timeToAcknowledgeReceipt* attribute is of type duration [XMLSCHEMA-2]. It specifies the  
1549 time period within which the receiving *Party* has to acknowledge receipt of a business document.

1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594

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

#### 8.4.13.10 *timeToAcknowledgeAcceptance* attribute

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

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

#### 8.4.13.11 *timeToPerform* attribute

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

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

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

#### 8.4.13.12 *retryCount* attribute

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

#### 8.4.14 *ChannelId* element

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

#### 8.4.15 *ActionContext* element

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

1595 Any business process/application implementation can use a combination of information in the  
1596 **action** attribute and the **ActionContext** elements to make message routing decisions. If using  
1597 alternative *Business-Collaboration* description schemas, the **action** attribute of the parent  
1598 **ThisPartyActionBinding** element and/or the [XML] **wildcard** element within the **ActionContext**  
1599 element MAY be used to make routing decisions above the level of the *Message Service*  
1600 *Handler*.

1601  
1602 The **ActionContext** element has the following elements:

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

1605  
1606 The **ActionContext** element also has the following attributes:

- 1607 • a REQUIRED **binaryCollaboration** attribute,
- 1608 • a REQUIRED **businessTransactionActivity** attribute,
- 1609 • a REQUIRED **requestOrResponseAction** attribute.

#### 1610 1611 **8.4.15.1 binaryCollaboration attribute**

1612 The REQUIRED **binaryCollaboration** attribute is a string that identifies the  
1613 **BinaryCollaboration** for which the parent **ThisPartyActionBinding** is defined. If the *Process-*  
1614 *Specification* document is defined by the ebXML Business Process Specification  
1615 Schema[ebBPSS], then the value of the **binaryCollaboration** attribute MUST match the value of  
1616 the **name** attribute of the **BinaryCollaboration** element as defined in the ebXML Business  
1617 Process Specification Schema[ebBPSS].

#### 1618 1619 **8.4.15.2 businessTransactionActivity attribute**

1620 The REQUIRED **businessTransactionActivity** attribute is a string that identifies the *Business*  
1621 *Transaction* for which the parent **ThisPartyActionBinding** is defined. If the *Process-*  
1622 *Specification* document is defined by the ebXML Business Process Specification  
1623 Schema[ebBPSS], the value of the **businessTransactionActivity** attribute MUST match the value  
1624 of the **name** attribute of the **BusinessTransactionActivity** element, whose parent is the **Binary**  
1625 **Collaboration** referred to by the **binaryCollaboration** attribute.

#### 1626 1627 **8.4.15.3 requestOrResponseAction attribute**

1628 The REQUIRED **requestOrResponseAction** attribute is a string that identifies either the  
1629 *Requesting or Responding Business Activity* for which the parent **ThisPartyActionBinding** is  
1630 defined. For a **ThisPartyActionBinding** defined for the request side of a message exchange, if  
1631 the *Process-Specification* document is defined by the ebXML Business Process Specification  
1632 Schema [ebBPSS], the value of the **requestOrResponseAction** attribute MUST match the value  
1633 of the **name** attribute of the **RequestingBusinessActivity** element corresponding to the  
1634 **BusinessTransaction** specified in the **businessTransactionActivity** attribute. Similarly, for the  
1635 response side of a message exchange, the value of the **requestOrResponseAction** attribute  
1636 MUST match the value of the **name** attribute of the **RespondingBusinessActivity** element  
1637 corresponding to the **BusinessTransaction** specified in the **businessTransactionActivity** attribute,  
1638 as defined in the ebXML Business Process Specification Schema[ebBPSS].

1639



#### 1640 **8.4.16 CollaborationActivity element**

1641 The *CollaborationActivity* element supports the ActionContext element by providing the ability  
 1642 to map any nested *BinaryCollaborations* as defined in the ebXML Business Process  
 1643 Specification Schema[ebBPSS] to the *action* attribute. The *CollaborationActivity* element  
 1644 MUST be present when the *Binary Collaboration* referred to by the *binaryCollaboration*  
 1645 attribute has a *Collaboration Activity* defined in the business process definition.

1646  
 1647 An example of the *CollaborationActivity* element is:

```
1648     <tp:CollaborationActivity
1649         tp:name="Credit Check"/>
```

1652 The *CollaborationActivity* element has zero or one child *CollaborationActivity* element to  
 1653 indicate further nesting of *Binary Collaborations*.

1655 The *CollaborationActivity* element also has one attribute:

- 1656 • a REQUIRED *name* attribute.

##### 1658 **8.4.16.1 name attribute**

1659 The REQUIRED *name* attribute is a string that identifies the *Collaboration Activity* included in  
 1660 the *Binary Collaboration*. If the *Process-Specification* document is defined by the ebXML  
 1661 Business Process Specification Schema[ebBPSS], the value of the *name* attribute MUST match  
 1662 the value of the *name* attribute of the *CollaborationActivity* within the *BinaryCollaboration*, as  
 1663 defined in the ebXML Business Process Specification Schema[ebBPSS].

#### 1665 **8.4.17 Certificate element**

1666 The *Certificate* element defines certificate information for use in this *CPP*. One or more  
 1667 *Certificate* elements MAY be provided for use in the various security functions in the *CPP*. An  
 1668 example of the *Certificate* element is:

```
1669     <tp:Certificate tp:certId="CompanyA_SigningCert">
1670         <ds:KeyInfo> . . . </ds:KeyInfo>
1671     </tp:Certificate>
```

1674 The *Certificate* element has a single REQUIRED attribute: *certId*. The *Certificate* element has a  
 1675 single child element: *ds:KeyInfo*.

1677 The *ds:KeyInfo* element may contain a complete chain of certificates, but the leaf certificate is  
 1678 the Certificate containing the key used in various asymmetric cryptographic operations. (The leaf  
 1679 certificate will be one that has been issued but has not issued certificates.) If the leaf certificate  
 1680 has been issued by an intermediate Certificate Authority, the complete chain to the root  
 1681 Certificate Authority SHOULD be included because it aids in testing certificate validity with  
 1682 respect to a set of trust anchors.

##### 1684 **8.4.17.1 certId attribute**

1685 The REQUIRED *certId* attribute is an [XML] ID that is referred to by a *CertificateRef* element

1686 elsewhere in the *CPP*. Here is an example of how a *CertificateRef* would refer to the *Certificate*  
 1687 element shown in the previous section:

```
1688
1689     <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
1690
```

#### 1691 **8.4.17.2 ds:KeyInfo element**

1692 The *ds:KeyInfo* element defines the certificate information. The content of this element and any  
 1693 subelements are defined by the XML Digital Signature specification[XMLDSIG].

1694  
 1695 NOTE: Software for creation of *CPPs* and *CPAs* MUST recognize the *ds:KeyInfo*  
 1696 element and insert the subelement structure necessary to define the certificate.  
 1697

#### 1698 **8.4.18 SecurityDetails element**

1699 The *SecurityDetails* element defines a set of *TrustAnchors* and an associated *SecurityPolicy* for  
 1700 use in this *CPP*. One or more *SecurityDetails* elements can be provided for use in the various  
 1701 security functions in the *CPP*. An example of the *SecurityDetails* element is:

```
1702
1703     <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
1704         <tp:TrustAnchors tp:trustId="MessageTrustAnchors">
1705             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
1706             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
1707         </tp:TrustAnchors>
1708         <tp:SecurityPolicy> ... </tp:SecurityPolicy>
1709     </tp:SecurityDetails>
1710
```

1711 The *SecurityDetails* element has zero or one *TrustAnchors* element that identify a set of  
 1712 certificates that are trusted by the *Party*. It also has zero or one *SecurityPolicy* element.

1713  
 1714 The *SecurityDetails* element allows agreement to be reached on what root certificates will be  
 1715 used in checking the validity of the other *Party's* certificates. It can also specify policy regarding  
 1716 operation of the public key infrastructure.

1717  
 1718 The *SecurityDetails* element has one attribute:

- 1719 • A REQUIRED *securityId* attribute.

#### 1720 1721 **8.4.18.1 securityId attribute**

1722 The REQUIRED *securityId* attribute is an [XML] ID that is referred to by an element elsewhere  
 1723 in the *CPP*. Here is an example of how a *SigningSecurityDetailsRef* would refer to the  
 1724 *SecurityDetails* element shown in the previous section:

```
1725
1726     <tp:SigningSecurityDetailsRef
1727 tp:securityId="CompanyA_MessageSecurity"/>
1728
```

#### 1729 **8.4.19 TrustAnchors element**

1730 The *TrustAnchors* element contains one or more *AnchorCertificateRef* elements, each of which  
 1731 refers to a *Certificate* element (under *PartyInfo*) that represents a certificate trusted by this

1732 *Party*. These trusted certificates are used in the process of certificate path validation. If a  
1733 certificate in question does not “chain” to one of this *Party*’s trust anchors, it is considered  
1734 invalid.

1735  
1736 The TrustAnchors element eventually resolves into XMLDsig *KeyInfo* elements. These elements  
1737 may contain several certificates (a chain), and may refer to those certificates using the  
1738 *RetrievalMethod* element. When there is a chain, the trust anchor is the “leaf” certificate with  
1739 respect to the “root” issuing Certificate Authority (CA) certificate. The root CA will be a self-  
1740 issued and self-signed certificate, and using the Issuer information and perhaps key usage  
1741 attributes, the leaf certificate (“issued but not issuing” within the chain) can be determined. The  
1742 chain is included for convenience in that validity checks typically will chain to a “root” CA.  
1743 Please note that the inclusion of a root CA in a chain does not mean that the root CA is being  
1744 announced as a trust anchor. It is possible for there to be a PKI policy in which some, but not all,  
1745 intermediate CAs are trusted. If a root CA were accepted as a trust anchor, all of its intermediate  
1746 CAs, and all the certificates they issue, would be validated. That might not be what was intended.

1747

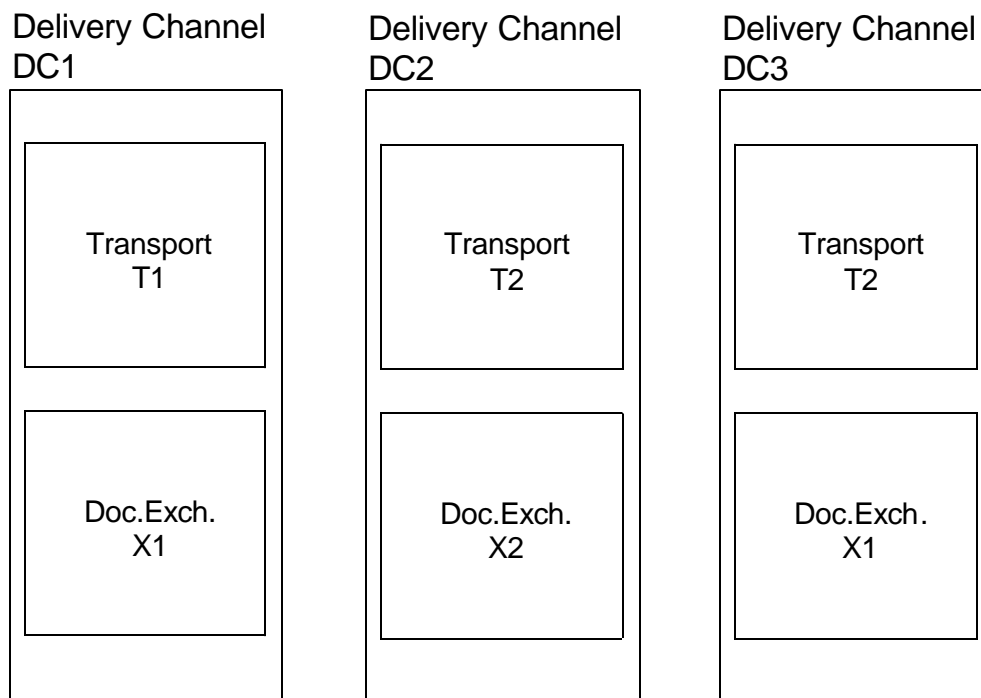
#### 1748 **8.4.20 SecurityPolicy element**

1749 The *SecurityPolicy* element is a placeholder for future apparatus that will enable the *Party* to  
1750 specify its policy and compliance regarding specific components of its public key infrastructure.  
1751 For example, it might stipulate revocation checking procedures or constraints related to name,  
1752 usage, or path length.

1753

#### 1754 **8.4.21 DeliveryChannel element**

1755 A delivery channel is a combination of a *Transport* element and a *DocExchange* element that  
1756 describes the *Party's Message* communication characteristics. The *CPP* SHALL contain one or  
1757 more *DeliveryChannel* elements, one or more *Transport* elements, and one or more  
1758 *DocExchange* elements. Each delivery channel SHALL refer to any combination of a  
1759 *DocExchange* element and a *Transport* element. The same *DocExchange* element or the same  
1760 *Transport* element can be referred to by more than one delivery channel. Two delivery  
1761 channels can use the same transport protocol and the same document-exchange protocol and  
1762 differ only in details such as communication addresses or security definitions. Figure 5 illustrates  
1763 three delivery channels.

**Figure 5: Three Delivery Channels**

1764  
 1765 The delivery channels have ID attributes with values "DC1", "DC2", and "DC3". Each delivery  
 1766 channel contains one transport definition and one document-exchange definition. Each transport  
 1767 definition and each document-exchange definition also has a name as shown in the figure. Note  
 1768 that delivery channel DC3 illustrates that a delivery channel can refer to the same transport  
 1769 definition and document-exchange definition used by other delivery channels but a different  
 1770 combination. In this case delivery channel DC3 is a combination of transport definition T2 (also  
 1771 referred to by delivery channel DC2) and document-exchange definition X1 (also referred to by  
 1772 delivery channel DC1).

1773  
 1774 A specific delivery channel SHALL be associated with each *PartyInfo* element,  
 1775 *OverrideMshActionBinding* element, *ServiceBinding* element, or *ThisPartyActionBinding*  
 1776 element (*action* attribute). Following is the delivery-channel syntax.

```

1777
1778 <tp:DeliveryChannel
1779     tp:channelId="channel1"
1780     tp:transportId="transport1"
1781     tp:docExchangeId="docExchange1"
1782     <tp:MessagingCharacteristics
1783         tp:syncReplyMode="none"
1784         tp:ackRequested="always"
1785         tp:ackSignatureRequested="always"
1786         tp:duplicateElimination="always"
1787         tp:actor="urn:oasis:names:tc:ebxml-msg:actor:nextMSH"/>
1788 </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,
- 1795 • a REQUIRED *transportId* attribute,
- 1796 • a REQUIRED *docExchangeId* attribute.

1797  
1798 The *DeliveryChannel* element has one REQUIRED child element, *MessagingCharacteristics*.

1799

#### 1800 **8.4.21.1 channelId attribute**

1801 The *channelId* attribute is an [XML] ID attribute that uniquely identifies the *DeliveryChannel*  
1802 element for reference, using IDREF attributes, from other parts of the *CPP* or *CPA*.

1803

#### 1804 **8.4.21.2 transportId attribute**

1805 The *transportId* attribute is an [XML] IDREF that identifies the *Transport* element that defines  
1806 the transport characteristics of the delivery channel. It MUST have a value that is equal to the  
1807 value of a *transportId* attribute of a *Transport* element elsewhere within the *CPP* document.

1808

#### 1809 **8.4.21.3 docExchangeId attribute**

1810 The *docExchangeId* attribute is an [XML] IDREF that identifies the *DocExchange* element that  
1811 defines the document-exchange characteristics of the delivery channel. It MUST have a value  
1812 that is equal to the value of a *docExchangeId* attribute of a *DocExchange* element elsewhere  
1813 within the *CPP* document.

1814

### 1815 **8.4.22 MessagingCharacteristics element**

1816 The *MessagingCharacteristics* element describes the attributes associated with messages  
1817 delivered over a given delivery channel. The collaborating parties can stipulate that these  
1818 attributes be fixed for all messages sent through the delivery channel, or they can agree that these  
1819 attributes are to be variable on a “per message” basis.

1820

1821 *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel  
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).

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

1867

1868 The value of "signalsAndResponse" indicates that the application will synchronously return the  
1869 *Business-response Message* in addition to one or more *Business* signals, plus any piggybacked  
1870 *MSH* level signals. In this case, each signal and response that is bundled into the same ebXML  
1871 message must appear as a separate MIME part (i.e., be placed in a separate payload container).  
1872 To be consistent, the ***TimeToAcknowledgeReceipt*** and **TimeToPerform** attributes under the  
1873 corresponding ***BusinessTransactionCharacteristics*** element **SHOULD** have identical values.  
1874 The ***timeToAcknowledgeAcceptance*** attribute, if specified, **SHOULD** also have the same value  
1875 as the above two timing attributes. The **Action** element in the synchronously returned ebXML  
1876 *Message* is determined by the name of the action in the *CPA* that corresponds to the appropriate  
1877 ***RespondingBusinessActivity*** in the *Business Process*.

1878

1879 The *Receipt Acknowledgment* signal for the *Business-response Message*, sent from the request  
1880 initiator back to the responder, if called for by the *Process-Specification*, **MUST** also be

1881 delivered over the same synchronous connection.

1882  
1883 NOTE: For HTTP 1.1 clients and servers, two HTTP requests and replies will have to be  
1884 sent and received on the same connection. Implementations that implicitly assume that a  
1885 HTTP connection will be closed after a single synchronous request reply interchange will  
1886 not be able to support the "signalsAndResponse" synchronous reply mode.

1887  
1888 The value of "none", which is the implied default value in the absence of the *syncReplyMode*  
1889 attribute, indicates that neither the *Business-response Message* nor any *Business* signal(s) will be  
1890 returned synchronously. In this case, all *Message Service Handler* level and *Business* level  
1891 messages will be returned as separate asynchronous messages.

1892  
1893 The ebXML *Message Service's SyncReply* element is included in the SOAP Header whenever  
1894 the *syncReplyMode* attribute has a value other than "none". If the delivery channel identifies a  
1895 transport protocol that has no synchronous capabilities (such as SMTP), the  
1896 *BusinessTransactionCharacteristics* element SHALL NOT have a *syncReplyMode* attribute  
1897 with a value other than "none".

1898  
1899 When the value of the *syncReplyMode* attribute is other than "none", a synchronous delivery  
1900 channel SHALL be used to exchange all messages necessary for conducting a business  
1901 transaction. If the *Process Specification* calls for the use of non-repudiation of receipt for the  
1902 response message, then the initiator is expected to return a signed *ReceiptAcknowledgment* signal  
1903 for the responder's response message.

#### 1904 1905 **8.4.22.2 ackRequested attribute**

1906 The IMPLIED *ackRequested* attribute is an enumeration comprised of the following possible  
1907 values:

- 1908 • "always"
- 1909 • "never"
- 1910 • "perMessage"

1911  
1912 This attribute has the default value "perMessage" meaning that the *AckRequested* element in the  
1913 SOAP Header is present or absent on a "per message" basis. If this attribute is set to "always",  
1914 then every message sent over the delivery channel MUST have an *AckRequested* element in the  
1915 SOAP Header. If this attribute is set to "never", then every message sent over the delivery  
1916 channel MUST NOT have an *AckRequested* element in the SOAP Header.

1917  
1918 If the *ackRequested* attribute is not set to "never", then the *ReliableMessaging* element must be  
1919 present under the corresponding *DocExchange* element to provide the necessary Reliable  
1920 Messaging parameters.

#### 1921 1922 **8.4.22.3 ackSignatureRequested attribute**

1923 The IMPLIED *ackSignatureRequested* attribute is an enumeration comprised of the following  
1924 possible values:

- 1925 • "always"
- 1926 • "never"

- 1927           • "perMessage"

1928  
1929 This attribute determines how the *signed* attribute within the *AckRequested* element in the SOAP  
1930 Header is to be set. It has the default value "perMessage" meaning that the *signed* attribute in the  
1931 *AckRequested* element within the SOAP Header is set to "true" or "false" on a "per message"  
1932 basis. If this attribute is set to "always", then every message sent over the delivery channel that  
1933 has an *AckRequested* element in the SOAP Header MUST have its *signed* attribute set to "true".  
1934 If this attribute is set to "never", then every message sent over the delivery channel that has an  
1935 *AckRequested* element in the SOAP Header MUST have its *signed* attribute set to "false". If the  
1936 *ackRequested* attribute is set to "never", the setting of the *ackSignatureRequested* attribute has  
1937 no effect.

1938  
1939 NOTE: By enabling the use of signed *Acknowledgment* for reliably delivered messages,  
1940 a weak form of non-repudiation of receipt can be supported. This is considered weaker  
1941 than the *Receipt Acknowledgment* signal because no schema check can be performed on  
1942 the payload prior to the return of the *Acknowledgment*. The *ackSignatureRequested*  
1943 attribute can be set independent of the value for the *isNonRepudiationReceiptRequired*  
1944 attribute under the *BusinessTransactionCharacteristics* element. Thus, even if the  
1945 original *Process-Specification* specifies that non-repudiation of receipt is to be  
1946 performed, the *CPP* and/or *CPA* can override this requirement, set  
1947 *isNonRepudiationReceiptRequired* to "false" and *ackSignatureRequested* to "always"  
1948 and thereby achieve the weak form of non-repudiation of receipt.

#### 1950 **8.4.22.4 duplicateElimination attribute**

1951 The IMPLIED *duplicateElimination* attribute is an enumeration comprised of the following  
1952 possible values:

- 1953           • "always"  
1954           • "never"  
1955           • "perMessage"

1956  
1957 This attribute determines whether the *DuplicateElimination* element within the *MessageHeader*  
1958 element in the SOAP Header is to be present. It has the default value "perMessage" meaning that  
1959 the *DuplicateElimination* element within the SOAP Header is present or absent on a "per  
1960 message" basis. If this attribute is set to "always", then every message sent over the delivery  
1961 channel MUST have a *DuplicateElimination* element in the SOAP Header. If this attribute is set  
1962 to "never", then every message sent over the delivery channel MUST NOT have a  
1963 *DuplicateElimination* element in the SOAP Header. If the *duplicateElimination* attribute is not  
1964 set to "never", then the *PersistDuration* element must be present under the corresponding  
1965 *DocExchange* element to provide the necessary persistent storage parameter.

#### 1967 **8.4.22.5 actor attribute**

1968 The IMPLIED *actor* attribute is an enumeration of the following possible values:

- 1969           • "urn:oasis:names:tc:ebxml-msg:actor:nextMSH"  
1970           • "urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"

1971  
1972 This is a URI that will be used as the value for the *actor* attribute in the *AckRequested* element



1973 in case the latter is present in the SOAP Header, as governed by the *ackRequested* attribute  
 1974 within the *MessagingCharacteristics* element in the *CPA*. If the *ackRequested* attribute is set to  
 1975 "never", the setting of the *actor* attribute has no effect.

1976

### 1977 8.4.23 Transport element

1978 The *Transport* element defines the *Party's* network communication capabilities. One or more  
 1979 *Transport* elements MUST be present in a *CPP*, each of which describes a mechanism the *Party*  
 1980 uses to send messages, a mechanism it uses to receive messages, or both. The following example  
 1981 illustrates the structure of a typical *Transport* element:

1982

```

1983 <tp:Transport tp:transportId="transportA1">
1984   <tp:TransportSender <!-- 0 or 1 times -->
1985     <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
1986     <tp:TransportClientSecurity>
1987       <tp:TransportSecurityProtocol tp:version="3.0">
1988         SSL
1989       </tp:TransportSecurityProtocol>
1990       <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
1991       <tp:ServerSecurityDetailsRef
1992         tp:securityId="CompanyA_TransportSecurity"/>
1993     </tp:TransportClientSecurity>
1994   </tp:TransportSender>
1995   <tp:TransportReceiver <!-- 0 or 1 times -->
1996     <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
1997     <tp:Endpoint
1998       tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler"
1999       tp:type="allPurpose"/>
2000     <tp:TransportServerSecurity>
2001       <tp:TransportSecurityProtocol tp:version="3.0">
2002         SSL
2003       </tp:TransportSecurityProtocol>
2004       <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
2005       <tp:ClientSecurityDetailsRef
2006         tp:securityId="CompanyA_TransportSecurity"/>
2007     </tp:TransportServerSecurity>
2008   </tp:TransportReceiver>
2009 </tp:Transport>
2010

```

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

2013

2014 A *Transport* that contains both *TransportSender* and *TransportReceiver* elements is said to be  
 2015 *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), its *CPP* MUST provide a *ServiceBinding* that contains *ActionBindings*  
 2018 that are bound to a *DeliveryChannel* that uses a bi-directional *Transport*.

2019

2020 A bi-directional *Transport* whose *TransportSender* and *TransportReceiver* elements use  
 2021 different transport protocols is said to be *asymmetric*. In a *CPA*, an asymmetric *Transport*  
 2022 offered by one *Party* MUST be matched with a complementary asymmetric *Transport* in the

2023 other *Party's CPP*. For example, a *Transport* composed of an HTTP sender and an SMTP  
2024 receiver would match with a *Transport* containing an SMTP sender and HTTP receiver.

2025

2026 NOTE: The ability of a transport to support bi-directional traffic does not imply that it  
2027 will be used for synchronous communication.

2028

2029 A *Transport* that contains either a *TransportSender* or a *TransportReceiver* element, but not  
2030 both, is said to be *unidirectional*. A unidirectional *Transport* can only be used for sending or  
2031 receiving messages (not both) depending on which element it includes.

2032

2033 A *CPP* contains as many *Transport* elements as are needed to fully express the *Party's* inbound  
2034 and outbound communication capabilities. If, for example, the *Party* can send and receive  
2035 messages via HTTP and SMTP, its *CPP* would contain a *Transport* element containing its HTTP  
2036 properties and another *Transport* element containing its SMTP properties.

2037

2038 The *Transport* element has

- 2039 • a REQUIRED *transportId* attribute

2040

#### 2041 8.4.23.1 transportId attribute

2042 The REQUIRED *transportId* attribute is an [XML] ID that is referred to by a *DeliveryChannel*  
2043 element elsewhere in the *CPP*. Here is an example of a *DeliveryChannel* that refers to the  
2044 *Transport* element shown in the previous section:

2045

```
2046 <tp:DeliveryChannel tp:channelId="channelA1"  
2047     tp:transportId="transportA1"  
2048     tp:docExchangeId="docExchangeA1">  
2049     <tp:BusinessTransactionCharacteristics . . . />  
2050     <tp:MessagingCharacteristics . . . />  
2051 </tp:DeliveryChannel>
```

2052

#### 2053 8.4.24 TransportSender element

2054 The *TransportSender* element contains properties related to the sending side of a  
2055 *DeliveryChannel*. Its REQUIRED *TransportProtocol* element specifies the transport protocol  
2056 that will be used for sending messages. The *AccessAuthentication* element(s), if present,  
2057 specifies the type(s) of access authentication supported by the client. The  
2058 *TransportClientSecurity* element, if present, defines the *Party's* provisions for client-side  
2059 transport layer security.

2060

2061 The *TransportSender* element has no attributes.

2062

#### 2063 8.4.25 TransportProtocol element

2064 The *TransportProtocol* element identifies a transport protocol that the *Party* is capable of using  
2065 to send or receive *Business* data. The IMPLIED *version* attribute identifies the specific version  
2066 of the protocol.

2067

2068 NOTE: It is the aim of this specification to enable support for any transport capable of  
 2069 carrying MIME content using the vocabulary defined herein.  
 2070

#### 2071 **8.4.26 AccessAuthentication element**

2072 The *AccessAuthentication* element, if present, indicates the authentication mechanism that MAY  
 2073 be used by a transport server to challenge a client request and by a client to provide  
 2074 authentication information to a server. For example, [RFC2617] specifies two access  
 2075 authentication schemes for HTTP: "basic" and "digest". A client that supports both would have  
 2076 two *AccessAuthentication* elements, as shown below. When multiple schemes are supported, the  
 2077 order in which they are specified in the CPP indicates the order of preference.  
 2078

```
2079 <tp:TransportSender>
2080 <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
2081 <tp:AccessAuthentication>digest</tp:AccessAuthentication>
2082 <tp:AccessAuthentication>basic</tp:AccessAuthentication>
2083 <tp:TransportClientSecurity>
2084     ...
2085 </tp:TransportClientSecurity>
2086 </tp:TransportSender>
```

2088 NOTE: A *CPA* will contain, for each *TransportSender* or *TransportReceiver*, only the  
 2089 agreed-upon *AccessAuthentication* elements.  
 2090

#### 2091 **8.4.27 TransportClientSecurity element**

2092 The *TransportClientSecurity* element provides information about this *Party's* transport client  
 2093 needed by the other *Party's* transport server to enable a secure connection to be established  
 2094 between the two. It contains a REQUIRED *TransportSecurityProtocol* element, zero or one  
 2095 *ClientCertificateRef* element, zero or one *ServerSecurityDetailsRef* element, and zero or one  
 2096 *EncryptionAlgorithm* element.  
 2097

2098 In asynchronous messaging mode, the sender will always be a client to the receiver's server. In  
 2099 synchronous messaging mode, the MSH-level reply (and maybe a bundled business signal and/or  
 2100 business response) is sent back over the same connection the initial business message arrived on.  
 2101 In such cases, where the sender is the server and the receiver is the client and the connection  
 2102 already exists, the sender's *TransportClientSecurity* and the receiver's *TransportServerSecurity*  
 2103 elements SHALL be ignored.  
 2104

#### 2105 **8.4.28 TransportSecurityProtocol element**

2106 The *TransportSecurityProtocol* element identifies the transport layer security protocol that is  
 2107 supported by the parent *Transport*. The IMPLIED *version* attribute identifies the specific version  
 2108 of the protocol.  
 2109

2110 For encryption, the protocol is TLS Version 1.0[RFC2246], which uses public-key encryption.  
 2111 Appendix E of the TLS Version 1.0 specification[RFC2246] covers backward compatibility with  
 2112 SSL [SSL].  
 2113

#### 2114 **8.4.29 ClientCertificateRef element**

2115 The *ClientCertificateRef* element identifies the certificate to be used by the client's transport  
2116 security module. The REQUIRED IDREF attribute *certId* identifies the certificate to be used by  
2117 referring to the *Certificate* element (under *PartyInfo*) that has the matching ID attribute value.  
2118 An SSL-capable HTTP client, for example, uses this certificate to authenticate itself with  
2119 receiver's secure HTTP server.

2120  
2121 The *ClientCertificateRef* element, if present, indicates that mutual authentication between client  
2122 and server (i.e., initiator and responder of the HTTP connection) MUST be performed.

2123  
2124 The *ClientCertificateRef* element has

- 2125 • A REQUIRED *certId* attribute

#### 2127 **8.4.30 ServerSecurityDetailsRef element**

2128 The *ServerSecurityDetailsRef* element identifies the trust anchors and security policy that this  
2129 *Party* will apply to the other *Party*'s server authentication certificate.

2130  
2131 The *ServerSecurityDetailsRef* element has

- 2132 • A REQUIRED *securityId* attribute

#### 2134 **8.4.31 Encryption Algorithm**

2135 Zero or more *EncryptionAlgorithm* elements may be included under the  
2136 *TransportClientSecurity* or *TransportServerSecurity* element. Multiple elements are of more  
2137 use in a CPP context, to announce capability or preferences; normally, a CPA will contain the  
2138 agreed upon context. When zero or more than one element is present in a CPA, the parties agree  
2139 to allow the automatic negotiation capability of the *TransportSecurityProtocol* to determine the  
2140 actual algorithm used.

2141  
2142 The elements' ordering will reflect the preference for algorithms. A primary reason for including  
2143 this element is to permit use of the *minimumStrength* attribute; a large value for this attribute  
2144 can indicate that high encryption strength is desired or has been agreed upon for the  
2145 *TransportSecurityProtocol*.

2146  
2147 See section 8.4.49 for the full description of this element.

2148  
2149 For SSL and TLS, it is customary to specify cipher suite values under the *EncryptionAlgorithm*  
2150 element. These values include, but are not limited to:

- 2151 • SSL\_RSA\_FIPS\_WITH\_3DES\_EDE\_CBC\_SHA,
- 2152 • TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA,
- 2153 • SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA,
- 2154 • SSL\_RSA\_WITH\_RC4\_128\_MD5,
- 2155 • SSL\_RSA\_WITH\_RC4\_128\_SHA,
- 2156 • SSL\_RSA\_WITH\_RC4\_128\_SHA,
- 2157 • SSL\_DH\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA,

- 2158 • SSL\_DHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA.

2159

2160 Consult the original specifications for enumerations and discussions of these values.

2161

#### 2162 **8.4.32 TransportReceiver element**

2163 The *TransportReceiver* element contains properties related to the receiving side of a  
2164 *DeliveryChannel*. Its REQUIRED *TransportProtocol* element specifies the transport protocol  
2165 that will be used for receiving messages. One or more REQUIRED *Endpoint* elements specify  
2166 logical addresses where messages can be received. The *AccessAuthentication* element, if  
2167 present, indicates the type(s) of access authentication supported by the server. Zero or one  
2168 *TransportServerSecurity* element defines the *Party*'s provisions for server-side transport layer  
2169 security.

2170

2171 The *TransportReceiver* element has no attributes.

2172

#### 2173 **8.4.33 Endpoint element**

2174 One or more *Endpoint* elements SHALL be provided for each *TransportReceiver* element. Each  
2175 *Endpoint* specifies a logical address and an indication of what kinds of messages can be received  
2176 at that location.

2177

2178 Each *Endpoint* has the following attributes:

2179

- a REQUIRED *uri* attribute

2180

- an IMPLIED *type* attribute

2181

##### 2182 **8.4.33.1 uri attribute**

2183 The REQUIRED *uri* attribute specifies a URI identifying the address of a resource. The value of  
2184 the *uri* attribute SHALL conform to the syntax for expressing URIs as defined in [RFC2396].

2185

##### 2186 **8.4.33.2 type attribute**

2187 The *type* attribute identifies the purpose of this endpoint. The value of *type* is an enumeration;  
2188 permissible values are "login", "request", "response", "error", and "allPurpose". There can be, at  
2189 most, one of each. If the *type* attribute is omitted, its value defaults to "allPurpose". The "login"  
2190 endpoint is used for the address for the initial *Message* between the two *Parties*. The "request"  
2191 and "response" endpoints are used for request and response *Messages*, respectively. To enable  
2192 error *Messages* to be received, each *Transport* element SHALL contain at least one endpoint of  
2193 type "error", "response", or "allPurpose".

2194

2195 The types of *Endpoint* within a *TransportReceiver* MUST not be overlapping. Thus, it would be  
2196 erroneous to include both an "allPurpose" *Endpoint* along with another *Endpoint* of any type.

2197

#### 2198 **8.4.34 TransportServerSecurity element**

2199 The *TransportServerSecurity* element provides information about this *Party*'s transport client  
2200 needed by the other *Party*'s transport server to enable a secure connection to be established  
2201 between the two. It contains a REQUIRED *TransportSecurityProtocol* element, a REQUIRED

2202 *ServerCertificateRef* element, zero or one *ClientSecurityDetailsRef* element, and zero or one  
2203 *EncryptionAlgorithm* element. See Section 8.4.31 for a description of the  
2204 *EncryptionAlgorithm* element.

2205  
2206 NOTE: See the note in Section 8.4.26 regarding the relevance of the  
2207 *TransportServerSecurity* element when synchronous replies are in use.  
2208

#### 2209 **8.4.35 ServerCertificateRef element**

2210 The *ServerCertificateRef* element, if present, identifies the certificate to be used by the server's  
2211 transport security module. The REQUIRED IDREF attribute *certId* identifies the certificate to be  
2212 used by referring to the *Certificate* element (under *PartyInfo*) that has the matching ID attribute  
2213 value. An SSL-enabled HTTP server, for example, uses this certificate to authenticate itself with  
2214 the sender's SSL client.

2215  
2216 The *ServerCertificateRef* element MUST be present if the transport security protocol uses  
2217 certificates. It MAY be omitted otherwise (e.g. if authentication is by password).  
2218

2219 The *ServerCertificateRef* element has

- 2220 • A REQUIRED *certId* attribute

2221

#### 2222 **8.4.36 ClientSecurityDetailsRef element**

2223 The *ClientSecurityDetailsRef* element, if present, identifies the trust anchors and security policy  
2224 that this *Party* will apply to the other *Party*'s client authentication certificate.

2225  
2226 The *ClientSecurityDetailsRef* element has

- 2227 • A REQUIRED *securityId* attribute

2228

#### 2229 **8.4.37 Transport protocols**

2230 In the following sections, we discuss the specific details of each supported transport protocol.

2231

##### 2232 **8.4.37.1 HTTP**

2233 HTTP is Hypertext Transfer Protocol[HTTP]. For HTTP, the endpoint is a URI that SHALL  
2234 conform to [RFC2396]. Depending on the application, there MAY be one or more endpoints,  
2235 whose use is determined by the application.

2236

2237 Following is an example of an HTTP endpoint:

2238

```
2239 <tp:Endpoint tp:uri="http://example.com/servlet/ebxmlhandler"  
2240 tp:type="request"/>
```

2241

2242 The "request" and "response" endpoints can be dynamically overridden for a particular request  
2243 or asynchronous response by application-specified URIs exchanged in *Business* documents  
2244 exchanged under the *CPA*.

2245

2246 For a synchronous response, the "response" endpoint is ignored if present. A synchronous  
2247 response is always returned on the existing connection, i.e. to the URI that is identified as the  
2248 source of the connection.

2249

#### 2250 **8.4.37.2 SMTP**

2251 SMTP is Simple Mail Transfer Protocol[SMTP]. For use with this standard, Multipurpose  
2252 Internet Mail Extensions[MIME] MUST be supported. For SMTP, the communication address is  
2253 the fully qualified mail address of the destination *Party* as defined by [RFC2822]. Following is  
2254 an example of an SMTP endpoint:

2255

```
2256 <tp:Endpoint tp:uri="mailto:ebxmlhandler@example.com"  
2257 tp:type="request" />
```

2258

2259 NOTE: The SMTP Mail Transfer Agent (MTA) can encode binary data when the receiving  
2260 MTA does not support binary transfer. In general, SMTP transfer may involve coding and  
2261 recoding of Content-Transfer-Encodings as a message moves along a sequence of MTAs. Such  
2262 changes can in some circumstances invalidate some kinds of signatures even though no  
2263 malicious actions or transmission errors have occurred.

2264

2265 NOTE: SMTP by itself (without any authentication or encryption) is subject to denial of service  
2266 and masquerading by unknown *Parties*. It is strongly suggested that those *Parties* who choose  
2267 SMTP as their transport layer also choose a suitable means of encryption and authentication  
2268 either in the document-exchange layer or in the transport layer such as [S/MIME].

2269

2270 NOTE: SMTP is an asynchronous protocol that does not guarantee a particular quality of service.  
2271 A transport-layer acknowledgment (i.e. an SMTP acknowledgment) to the receipt of a mail  
2272 *Message* constitutes an assertion on the part of the SMTP server that it knows how to deliver the  
2273 mail *Message* and will attempt to do so at some point in the future. However, the *Message* is not  
2274 hardened and might never be delivered to the recipient. Furthermore, the sender will see a  
2275 transport-layer acknowledgment only from the nearest node. If the *Message* passes through  
2276 intermediate nodes, SMTP does not provide an end-to-end acknowledgment. Therefore receipt  
2277 of an SMTP acknowledgement does not guarantee that the *Message* will be delivered to the  
2278 application and failure to receive an SMTP acknowledgment is not evidence that the *Message*  
2279 was not delivered. It is RECOMMENDED that the reliable-messaging protocol in the ebXML  
2280 *Message Service* be used with SMTP.

2281

#### 2282 **8.4.37.3 FTP**

2283 FTP is File Transfer Protocol[RFC959].

2284

2285 Each *Party* sends a *Message* using FTP PUT. The endpoint specifies the user id and input  
2286 directory path (for PUTs to this *Party*). An example of an FTP endpoint is:

2287

```
2288 <tp:Endpoint uri="ftp://userid@server.foo.com"  
2289 tp:type="request" />
```

2290

2291 Since FTP needs to be compatible across all implementations, the FTP for ebXML will use the  
2292 minimum sets of commands and parameters available for FTP as specified in [RFC959], Section

2293 5.1, and modified in [RFC1123], Section 4.1.2.13. The mode SHALL be stream only and the  
2294 type MUST be ASCII Non-print (AN), Image (I) (binary), or Local 8 (L 8) (binary between 8-bit  
2295 machines and machines with 36 bit words – for an 8-bit machine Local 8 is the same as Image).

2296  
2297 Stream mode closes the data connection upon end of file. The server side FTP MUST set control  
2298 to "PASV" before each transfer command to obtain a unique port pair if there are multiple third  
2299 party sessions.

2300  
2301 NOTE: [RFC 959] states that User-FTP SHOULD send a PORT command to assign a  
2302 non-default data port before each transfer command is issued to allow multiple transfers  
2303 during a single FTP because of the long delay after a TCP connection is closed until its  
2304 socket pair can be reused.

2305  
2306 NOTE: The format of the 227 reply to a PASV command is not well standardized and an  
2307 FTP client might assume that the parentheses indicated in [RFC959] will be present when  
2308 in some cases they are not. If the User-FTP program doesn't scan the reply for the first  
2309 digit of host and port numbers, the result will be that the User-FTP might point at the  
2310 wrong host. In the response, the h1, h2, h3, h4 is the IP address of the server host and the  
2311 p1, p2 is a non-default data transfer port that PASV has assigned.

2312  
2313 NOTE: As a recommendation for firewall transparency, [RFC1579] proposes that the  
2314 client sends a PASV command, allowing the server to do a passive TCP open on some  
2315 random port, and inform the client of the port number. The client can then do an active  
2316 open to establish the connection.

2317  
2318 NOTE: Since STREAM mode closes the data connection upon end of file, the receiving  
2319 FTP might assume abnormal disconnect if a 226 or 250 control code hasn't been received  
2320 from the sending machine.

2321  
2322 NOTE: [RFC1579] also makes the observation that it might be worthwhile to enhance the  
2323 FTP protocol to have the client send a new command APSV (all passive) at startup that  
2324 would allow a server that implements this option to always perform a passive open. A  
2325 new reply code 151 would be issued in response to all file transfer requests not preceded  
2326 by a PORT or PASV command; this *Message* would contain the port number to use for  
2327 that transfer. A PORT command could still be sent to a server that had previously  
2328 received APSV; that would override the default behavior for the next transfer operation,  
2329 thus permitting third-party transfers.

2330

#### 2331 **8.4.38 DocExchange Element**

2332 The *DocExchange* element provides information that the *Parties* MUST agree on regarding  
2333 exchange of documents between them. This information includes the messaging service  
2334 properties (e.g. ebXML *Message Service*[ebMS]).

2335

2336 Following is the structure of the *DocExchange* element of the *CPP*. Subsequent sections  
2337 describe each child element in greater detail.



```

2338
2339     <tp:DocExchange tp:docExchangeId="docExchangeB1">
2340         <tp:ebXMLSenderBinding tp:version="1.1"> <!-- 0 or 1 -->
2341             <tp:ReliableMessaging> <!-- 0 or 1 -->
2342                 . . .
2343             </tp:ReliableMessaging>
2344             <tp:PersistDuration> <!-- 0 or 1 -->
2345                 . . .
2346             </tp:PersistDuration>
2347             <tp:SenderNonRepudiation> <!-- 0 or 1 -->
2348                 . . .
2349             </tp:SenderNonRepudiation>
2350             <tp:SenderDigitalEnvelope> <!-- 0 or 1 -->
2351                 . . .
2352             </tp:SenderDigitalEnvelope>
2353             <tp:NamespaceSupported> <!-- 0 or more -->
2354                 . . .
2355             </tp:NamespaceSupported>
2356         </tp:ebXMLSenderBinding>
2357         <tp:ebXMLReceiverBinding tp:version="1.1"> <!-- 0 or 1 -->
2358             <tp:ReliableMessaging> <!-- 0 or 1 -->
2359                 . . .
2360             </tp:ReliableMessaging>
2361             <tp:PersistDuration> <!-- 0 or 1 -->
2362                 . . .
2363             </tp:PersistDuration>
2364             <tp:ReceiverNonRepudiation> <!-- 0 or 1 -->
2365                 . . .
2366             </tp:ReceiverNonRepudiation>
2367             <tp:ReceiverDigitalEnvelope> <!-- 0 or 1 -->
2368                 . . .
2369             </tp:ReceiverDigitalEnvelope>
2370             <tp:NamespaceSupported> <!-- 0 or more -->
2371                 . . .
2372             </tp:NamespaceSupported>
2373         </tp:ebXMLReceiverBinding>
2374     </tp:DocExchange>

```

2375  
2376 The *DocExchange* element is comprised of zero or one *ebXMLSenderBinding* element and zero  
2377 or one *ebXMLReceiverBinding* 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  
2383 *XMLPSenderBinding/XMLPReceiverBinding*, which might define support for the future  
2384 XML Protocol specification.

#### 2385 8.4.38.1 docExchangeId attribute

2386 The *DocExchange* element has a single REQUIRED *docExchangeId* attribute that is an [XML]  
2387 ID that provides a unique identifier that can be referenced from elsewhere within the *CPP*  
2388 document.

2390

### 2391 8.4.39 ebXMLSenderBinding element

2392 The *ebXMLSenderBinding* element describes properties related to sending messages with the  
2393 ebXML *Message Service*[ebMS]. The *ebXMLSenderBinding* element is comprised of the  
2394 following child elements:

- 2395 • zero or one *ReliableMessaging* element which specifies the characteristics of reliable  
2396 messaging,
- 2397 • zero or one *PersistDuration* element which specifies the duration for which certain  
2398 messages have to be stored persistently for the purpose of duplicate elimination,
- 2399 • zero or one *SenderNonRepudiation* element which specifies the sender's  
2400 requirements and certificate for message signing,
- 2401 • zero or one *SenderDigitalEnvelope* element which specifies the sender's  
2402 requirements for encryption by the digital-envelope[DIGENV] method,
- 2403 • zero or more *NamespaceSupported* elements that identify any namespace extensions  
2404 supported by the messaging service implementation.

2405  
2406 The *ebXMLSenderBinding* element has one attribute:

- 2407 • a REQUIRED *version* attribute

#### 2408 8.4.39.1 version attribute

2409 The REQUIRED *version* attribute identifies the version of the ebXML *Message Service*  
2410 specification being used.  
2411  
2412

### 2413 8.4.40 ReliableMessaging element

2414 The *ReliableMessaging* element specifies the properties of reliable ebXML *Message* exchange.  
2415 The default that applies if the *ReliableMessaging* element is omitted is "BestEffort". The  
2416 following is the element structure:

```
2417 <tp:ReliableMessaging>
2418   <tp:Retries>5</tp:Retries>
2419   <tp:RetryInterval>PT2H</tp:RetryInterval>
2420   <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
2421 </tp:ReliableMessaging>
```

2422  
2423  
2424 The *ReliableMessaging* element is comprised of the following child elements.

- 2425 • zero or one *Retries* element,
- 2426 • zero or one *RetryInterval* element,
- 2427 • a REQUIRED *MessageOrderSemantics* element.

#### 2428 8.4.40.1 Retries and RetryInterval elements

2429  
2430 The *Retries* and *RetryInterval* elements specify the permitted number of retries and the interval,  
2431 expressed as an XML Schema[XMLSCHEMA-2] duration, between retries of sending a reliably  
2432 delivered *Message* following a timeout waiting for the *Acknowledgment*. The purpose of the  
2433 *RetryInterval* element is to improve the likelihood of success on retry by deferring the retry until  
2434 any temporary conditions that caused the error might be corrected. The *RetryInterval* applies to  
2435 the time between sending of the original message and the first retry, as well as the time between  
2436

2437 all subsequent retries.

2438  
2439 The *Retries* and *RetryInterval* elements MUST either be included together or be omitted  
2440 together. If they are omitted, the values of the corresponding quantities (number of retries and  
2441 retry interval) are a local matter at each *Party*.

2442

#### 2443 8.4.40.2 MessageOrderSemantics element

2444 The *MessageOrderSemantics* element is an enumeration comprised of the following possible  
2445 values:

- 2446 • "Guaranteed"
- 2447 • "NotGuaranteed"

2448

2449 The presence of a *MessageOrder* element in the SOAP Header for ebXML messages determines  
2450 if the ordering of messages sent from the *From Party* needs to be preserved so that the *To Party*  
2451 receives those messages in the order in which they were sent. If the *MessageOrderSemantics*  
2452 element is set to "Guaranteed", then the ebXML message MUST contain a *MessageOrder*  
2453 element in the SOAP Header. If the *MessageOrderSemantics* element is set to "NotGuaranteed",  
2454 then the ebXML message MUST NOT contain a *MessageOrder* element in the SOAP Header.  
2455 Guaranteed message ordering implies the use of duplicate elimination. Therefore, the  
2456 PersistDuration element must also appear if MessageOrderSemantics is set to "Guaranteed".

2457

#### 2458 8.4.41 PersistDuration element

2459 The value of the *PersistDuration* element is the minimum length of time, expressed as an XML  
2460 Schema[XMLSCHEMA-2] duration, that data from a *Message* that is sent reliably is kept in  
2461 *Persistent Storage* by an ebXML *Message-Service* implementation that receives that *Message* to  
2462 facilitate the elimination of duplicates. This duration also applies to response messages that are  
2463 kept persistently to allow automatic replies to duplicate messages without their repeated  
2464 processing by the application.

2465

#### 2466 8.4.42 SenderNonRepudiation element

2467 The *SenderNonRepudiation element* conveys the message sender's requirements and certificate  
2468 for non-repudiation. Non-repudiation both proves who sent a *Message* and prevents later  
2469 repudiation of the contents of the *Message*. Non-repudiation is based on signing the *Message*  
2470 using XML Digital Signature[XMLDSIG]. The element structure is as follows:

2471

```

2472 <tp:SenderNonRepudiation>
2473   <tp:NonRepudiationProtocol>
2474     http://www.w3.org/2000/09/xmldsig#
2475   </tp:NonRepudiationProtocol>
2476   <tp:HashFunction>
2477     http://www.w3.org/2000/09/xmldsig#sha1
2478   </tp:HashFunction>
2479   <tp:SignatureAlgorithm>
2480     http://www.w3.org/2000/09/xmldsig#dsa-sha1
2481   </tp:SignatureAlgorithm>
2482   <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert" />
2483 </tp:SenderNonRepudiation>

```

2484  
2485 If the *SenderNonRepudiation* element is omitted, the *Messages* are not digitally signed.

2486  
2487 The *SenderNonRepudiation* element is comprised of the following child elements:

- 2488 • a REQUIRED *NonRepudiationProtocol* element,
- 2489 • a REQUIRED *HashFunction* (e.g. SHA1, MD5) element,
- 2490 • a REQUIRED *SignatureAlgorithm* element,
- 2491 • a REQUIRED *SigningCertificateRef* element

2492

#### 2493 **8.4.43 NonRepudiationProtocol element**

2494 The REQUIRED *NonRepudiationProtocol* element identifies the technology that will be used to  
2495 digitally sign a *Message*. It has a single IMPLIED *version* attribute whose value is a string that  
2496 identifies the version of the specified technology.

2497

#### 2498 **8.4.44 HashFunction element**

2499 The REQUIRED *HashFunction* element identifies the algorithm that is used to compute the  
2500 digest of the *Message* being signed.

2501

#### 2502 **8.4.45 SignatureAlgorithm element**

2503 The REQUIRED *SignatureAlgorithm* element identifies the algorithm that is used to compute  
2504 the value of the digital signature. Expected values include: RSA-MD5, RSA-SHA1, DSA-MD5,  
2505 DSA-SHA1, SHA1withRSA, MD5withRSA, and so on.

2506

2507 NOTE: Implementations should be prepared for values in either case and with varying  
2508 usage of hyphens and conjunctions.

2509

2510 The *SignatureAlgorithm* element has three attributes:

- 2511 • an IMPLIED *oid* attribute,
- 2512 • an IMPLIED *w3c* attribute,
- 2513 • an IMPLIED *enumeratedType* attribute.

2514

##### 2515 **8.4.45.1 oid attribute**

2516 The *oid* attribute serves as a way to supply an object identifier for the signature algorithm. The  
2517 formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the  
2518 assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D.

2519 Commonly used values (in the IETF dotted integer format) for signature algorithms include:

- 2520 • 1.2.840.113549.1.1.4 - MD5 with RSA encryption,
- 2521 • 1.2.840.113549.1.1.5 - SHA-1 with RSA Encryption.

2522

##### 2523 **8.4.45.2 w3c attribute**

2524 The *w3c* attribute serves as a way to supply an object identifier for the signature algorithm. The  
2525 definitions of these values are found in the [XMLDSIG] or [XMLENC] specifications. Expected  
2526 values for signature algorithms include:

- 2527       • <http://www.w3.org/2000/09/xmldsig#dsa-sha1>,
- 2528       • <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

2529

#### 2530 **8.4.45.3 enumeratedType attribute**

2531 The *enumeratedType* attribute specifies a different way of interpreting the text value of the  
 2532 element *SignatureElement*. This attribute is for identifying future signature algorithm  
 2533 identification schemes and formats.

2534

#### 2535 **8.4.46 SigningCertificateRef element**

2536 The REQUIRED *SigningCertificateRef* element identifies the certificate the sender uses for  
 2537 signing messages. Its REQUIRED IDREF attribute, *certId* refers to the *Certificate* element  
 2538 (under *PartyInfo*) that has the matching ID attribute value.

2539

#### 2540 **8.4.47 SenderDigitalEnvelope element**

2541 The *SenderDigitalEnvelope* element provides the sender's requirements for message encryption  
 2542 using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in which the  
 2543 *Message* is encrypted by symmetric encryption (shared secret key) and the secret key is sent to  
 2544 the *Message* recipient encrypted with the recipient's public key. The element structure is:

2545

```
2546     <tp:SenderDigitalEnvelope>
2547         <tp:DigitalEnvelopeProtocol tp:version="2.0">
2548             S/MIME
2549         </tp:DigitalEnvelopeProtocol>
2550         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
2551         <tp:EncryptionSecurityDetailsRef
2552             tp:securityId="CompanyA_MessageSecurity"/>
2553     </tp:SenderDigitalEnvelope>
```

2554

2555 The *SenderDigitalEnvelope* element contains

- 2556       • a REQUIRED *DigitalEnvelopeProtocol* element,
- 2557       • a REQUIRED *EncryptionAlgorithm* element
- 2558       • zero or one *EncryptionSecurityDetailsRef* element.

2559

#### 2560 **8.4.48 DigitalEnvelopeProtocol element**

2561 The REQUIRED *DigitalEnvelopeProtocol* element identifies the message encryption protocol to  
 2562 be used. The REQUIRED *version* attribute identifies the version of the protocol.

2563

#### 2564 **8.4.49 EncryptionAlgorithm element**

2565 The REQUIRED *EncryptionAlgorithm* element identifies the encryption algorithm to be used.

2566

2567 The *EncryptionAlgorithm* element has four attributes:

- 2568       • an IMPLIED *minimumStrength* attribute,
- 2569       • an IMPLIED *oid* attribute,
- 2570       • an IMPLIED *w3c* attribute,

- 2571           • an IMPLIED *enumeratedType* attribute.

2572

#### 2573 **8.4.49.1 minimumStrength attribute**

2574 The *minimumStrength* attribute describes the effective strength the encryption algorithm must  
2575 provide in terms of “effective” or random bits. This value may be less than the key length in bits  
2576 when check bits are used in the key. So, for example, the 8 check bits of a 64-bit DES key would  
2577 not be included in the count, and to require a minimum strength the same as that supplied by  
2578 DES would be reported by setting *minimumStrength* to 56.

2579

#### 2580 **8.4.49.2 oid attribute**

2581 The *oid* attribute serves as a way to supply an object identifier for the encryption algorithm. The  
2582 formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the  
2583 assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D.

2584 Commonly used values (in the IETF dotted integer format) for encryption algorithms include:

- 2585           • 1.2.840.113549.3.2 (RC2-CBC), 1.2.840.113549.3.4 (RC4 Encryption Algorithm ),  
2586           • 1.2.840.113549.3.7 (DES-EDE3-CBC ), 1.2.840.113549.3.9 (RC5 CBC Pad),  
2587           • 1.2.840.113549.3.10 (DES CDMF), 1.2.840, 1.3.14.3.2.7 (DES-CBC).

2588

#### 2589 **8.4.49.3 w3c attribute**

2590 The *w3c* attribute serves as a way to supply an object identifier for the encryption algorithm. The  
2591 definitions of these values are in the [XMLENC] specification. Expected values include:

- 2592           • <http://www.w3.org/2001/04/xmlenc#3des-cbc>,  
2593           • <http://www.w3.org/2001/04/xmlenc#aes128-cbc>,  
2594           • [http://www.w3.org/2001/04/xmlenc - aes256-cbc](http://www.w3.org/2001/04/xmlenc-aes256-cbc).

2595

#### 2596 **8.4.49.4 enumeratedTypeAttribute**

2597 The *enumeratedType* attribute specifies a way of interpreting the text value of the  
2598 *EncryptionAlgorithm* element. This attribute is for identifying future algorithm identification  
2599 schemes and formats.

2600

#### 2601 **8.4.50 EncryptionSecurityDetailsRef element**

2602 The *EncryptionSecurityDetailsRef* element identifies the trust anchors and security policy that  
2603 this (sending) *Party* will apply to the other (receiving) *Party*'s encryption certificate. Its  
2604 REQUIRED IDREF attribute, *securityId*, refers to the *SecurityDetails* element (under *PartyInfo*)  
2605 that has the matching ID attribute value.

2606

#### 2607 **8.4.51 NamespaceSupported element**

2608 The *NamespaceSupported* element identifies the namespaces supported by the messaging  
2609 service implementation. It has a REQUIRED *location* attribute and an IMPLIED *version*  
2610 attribute. While the *NamespaceSupported* element can be used to list the namespaces that could  
2611 be expected to be used during document exchange, the motivation is primarily for extensions,  
2612 version variants, and other enhancements that might not be expected, or have only recently  
2613 emerged into use.

2614

2615 For example, support for Security Assertion Markup Language[SAML] would be defined as  
 2616 follows:

```
2617
2618     <tp:NamespaceSupported
2619     tp:location="http://www.oasis-open.org/committees/security/docs/draft-
2620     sstc-schema-assertion-27.xsd" tp:version="1.0">
2621     http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-
2622     assertion-27.xsd</tp:NamespaceSupported>
```

2623  
 2624 In addition, the *NamespaceSupported* element can be used to identify the namespaces associated  
 2625 with the message body parts (see Section 8.5), and especially when these namespaces are not  
 2626 implicitly indicated through parts of the *ProcessSpecification* or when they indicate extensions  
 2627 of namespaces for payload body parts.

2628

#### 2629 **8.4.52 ebXMLReceiverBinding element**

2630 The *ebXMLReceiverBinding* element describes properties related to receiving messages with the  
 2631 ebXML *Message Service*[ebMS]. The *ebXMLReceiverBinding* element is comprised of the  
 2632 following child elements:

- 2633 • zero or one *ReliableMessaging* element (see Section 8.4.40),
- 2634 • zero or one *ReceiverNonRepudiation* element which specifies the receiver's  
 2635 requirements for message signing,
- 2636 • zero or one *ReceiverDigitalEnvelope* element which specifies the receiver's  
 2637 requirements and certificate for encryption by the digital-envelope[DIGENV]  
 2638 method,
- 2639 • zero or more *NamespaceSupported* elements (see Section 8.4.51).

2640

2641 The *ebXMLReceiverBinding* element has one attribute:

- 2642 • a REQUIRED *version* attribute (see Section 8.4.39.1)

2643

#### 2644 **8.4.53 ReceiverNonRepudiation element**

2645 The *ReceiverNonRepudiation element* conveys the message receiver's requirements for non-  
 2646 repudiation. Non-repudiation both proves who sent a *Message* and prevents later repudiation of  
 2647 the contents of the *Message*. Non-repudiation is based on signing the *Message* using XML  
 2648 Digital Signature[XMLDSIG]. The element structure is as follows:

```
2649
2650     <tp:ReceiverNonRepudiation>
2651         <tp:NonRepudiationProtocol>
2652             http://www.w3.org/2000/09/xmlldsig#
2653         </tp:NonRepudiationProtocol>
2654         <tp:HashFunction>
2655             http://www.w3.org/2000/09/xmlldsig#sha1
2656         </tp:HashFunction>
2657         <tp:SignatureAlgorithm>
2658             http://www.w3.org/2000/09/xmlldsig#dsa-sha1
2659         </tp:SignatureAlgorithm>
2660         <tp:SigningSecurityDetailsRef
2661         tp:certId="CompanyA_MessageSecurity"/>
```

2662           </tp:ReceiverNonRepudiation>

2663

2664 If the *ReceiverNonRepudiation* element is omitted, the *Messages* are not digitally signed.

2665

2666 The *ReceiverNonRepudiation* element is comprised of the following child elements:

- 2667       • a REQUIRED *NonRepudiationProtocol* element (see Section 8.4.43),
- 2668       • a REQUIRED *HashFunction* (e.g. SHA1, MD5) element (see Section 8.4.44),
- 2669       • a REQUIRED *SignatureAlgorithm* element (see Section 8.4.45),
- 2670       • zero or one *SigningSecurityDetailsRef* element

2671

#### 2672 **8.4.54 SigningSecurityDetailsRef element**

2673 The *SigningSecurityDetailsRef* element identifies the trust anchors and security policy that this  
 2674 (receiving) *Party* will apply to the other (sending) *Party*'s signing certificate. Its REQUIRED  
 2675 IDREF attribute, *securityId*, refers to the *SecurityDetails* element (under *PartyInfo*) that has the  
 2676 matching ID attribute value.

2677

#### 2678 **8.4.55 ReceiverDigitalEnvelope element**

2679 The *ReceiverDigitalEnvelope* element provides the receiver's requirements for message  
 2680 encryption using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in  
 2681 which the *Message* is encrypted by symmetric encryption (shared secret key) and the secret key  
 2682 is sent to the *Message* recipient encrypted with the recipient's public key. The element structure  
 2683 is:

2684

```

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

2694 The *ReceiverDigitalEnvelope* element contains

- 2695       • a REQUIRED *DigitalEnvelopeProtocol* element (see Section 8.4.48),
- 2696       • a REQUIRED *EncryptionAlgorithm* element (see Section 8.4.49),
- 2697       • a REQUIRED *EncryptionCertificateRef* element.

2698

#### 2699 **8.4.56 EncryptionCertificateRef element**

2700 The REQUIRED *EncryptionCertificateRef* element identifies the certificate the sender uses for  
 2701 encrypting messages. Its REQUIRED IDREF attribute, *certId* refers to the *Certificate* element  
 2702 (under *PartyInfo*) that has the matching ID attribute value.

2703 .

#### 2704 **8.4.57 OverrideMshActionBinding element**

2705 The *OverrideMshActionBinding* element can occur zero or more times. It has two REQUIRED



2706 attributes. The *action* attribute identifies the *Message Service Handler* level action whose  
2707 delivery is not to use the default *DeliveryChannel* for *Message Service Handler* actions. The  
2708 channelId attribute specifies the *DeliveryChannel* to be used instead.

2709

## 2710 8.5 SimplePart element

2711 The *SimplePart* element provides a repeatable list of the constituent parts, primarily identified by  
2712 the MIME content-type value. The *SimplePart* element has two REQUIRED attributes: *id* and  
2713 *mimetype*. The *id* attribute, of type ID, provides the value that will be used later to reference this  
2714 *Message* part when specifying how the parts are packaged into composites, if composite  
2715 packaging is present. The *mimetype* attribute can provide actual values of content-type for the  
2716 simple *Message* part being specified. The attribute's values may also make use of an asterisk  
2717 wildcard, "\*", to indicate either an arbitrary top-level type, an arbitrary sub-type, or a completely  
2718 arbitrary type, "\*/\*". SimpleParts with wildcards in types can be used in indicating more open  
2719 packaging processing capabilities.

2720

2721 *SimplePart* also has an IMPLIED *xlink:role* attribute which identifies some resource that  
2722 describes the mime part or its purpose. If present, then it SHALL have a value that is a valid URI  
2723 in accordance with the [XLINK] specification. The following are examples of *SimplePart*  
2724 elements:

2725

```
2726 <tp:SimplePart tp:id="I001" tp:mimetype="text/xml"/>  
2727 <tp:SimplePart tp:id="I002" tp:mimetype="application/xml"/>  
2728 <tp:SimplePart tp:id="I002" tp:mimetype="*/xml"/>
```

2729

2730 The *SimplePart* element can have zero or more *NamespaceSupported* elements. Each of these  
2731 identifies any namespace supported for the XML that is packaged in the parent simple body part.

2732

2733 The context of *Packaging* can very easily render it pointless to list all the namespaces used in a  
2734 *SimplePart*. For example, when defining the *SimplePart* for a SOAP envelope, as part of an  
2735 ebXML Message, it is not necessary to list all the namespaces. If, however, any unusual  
2736 extensions, new versions, or unusual security extensions are present, it is useful to announce  
2737 these departures explicitly in the packaging. It is not, however, incorrect to list all namespaces  
2738 used in a *SimplePart*, even where these namespaces have been mandated by a given messaging  
2739 protocol. By convention, when a full listing of namespaces is supplied within a *SimplePart*  
2740 element, the first *NamespaceSupported* element identifies the schema for the *SimplePart* while  
2741 subsequent *NamespaceSupported* elements represent namespaces that are imported by that  
2742 schema. Any additional *NamespaceSupported* elements indicate extensions.

2743

2744 NOTE: The explicit identification of imported namespaces is discretionary. Thus, the  
2745 CPP and CPA examples in Appendix A and Appendix B explicitly identify the ebXML  
2746 Messaging Service namespace but omit the SOAP envelope and XML Digital Signature  
2747 namespaces that are imported into the schema for the ebXML Messaging Service  
2748 namespace.

2749

2750 The same *SimplePart* element can be referenced from (i.e., reused in) multiple *Packaging*  
2751 elements.

2752

2753 **8.6 Packaging element**

2754 The subtree of the *Packaging* element provides specific information about how the *Message*  
 2755 *Header* and payload constituent(s) are packaged for transmittal over the transport, including the  
 2756 crucial information about what document-level security packaging is used and the way in which  
 2757 security features have been applied. Typically the subtree under the *Packaging* element indicates  
 2758 the specific way in which constituent parts of the *Message* are organized. MIME processing  
 2759 capabilities are typically the capabilities or agreements described in this subtree. The *Packaging*  
 2760 element provides information about MIME content types, XML namespaces, security  
 2761 parameters, and MIME structure of the data that is exchanged between *Parties*.

2762

2763 The following is an example of a *Packaging* element which references the example *SimplePart*  
 2764 elements given in Section 8.5:

2765

```

2766     <!-- Simple ebXML S/MIME Packaging for application-based payload
2767           encryption -->
2768     <tp:Packaging>
2769       <tp:ProcessingCapabilities tp:generate="true" tp:parse="true"/>
2770       <tp:CompositeList>
2771         <tp:Encapsulation
2772           <!-- I002 is the payload being encrypted -->
2773           tp:id="I003"
2774           tp:mimetype="application/pkcs7-mime"
2775           tp:mimeparameters="smime-type=&quot;enveloped-data&quot;">
2776             <Constituent tp:idref="I002"/>
2777         </tp:Encapsulation>
2778         <tp:Composite tp:id="I004"
2779           <!-- I001 is the SOAP envelope. The ebXML message is made
2780             up of the SOAP envelope and the encrypted payload. -->
2781           tp:mimetype="multipart/related"
2782           tp:mimeparameters="type=&quot;text/xml&quot;;
2783             version=&quot;1.0&quot;">
2784             <tp:Constituent tp:idref="I001"/>
2785             <tp:Constituent tp:idref="I003"/>
2786           </tp:Composite>
2787         </tp:CompositeList>
2788     </tp:Packaging>
  
```

2789

2790 The *Packaging* element has one attribute; the REQUIRED *id* attribute, with type ID. It is  
 2791 referred to in the *ThisPartyActionBinding* element, by using the IDREF attribute, *packageId*.

2792

2793 The child elements of the *Packaging* element are *ProcessingCapabilities* and *CompositeList*.  
 2794 This set of elements can appear one or more times as a child of each *Packaging* element.

2795

2796 **8.6.1 ProcessingCapabilities element**

2797 The *ProcessingCapabilities* element has two REQUIRED attributes with Boolean values of  
 2798 either "true" or "false". The attributes are *parse* and *generate*. Normally, these attributes will  
 2799 both have values of "true" to indicate that the packaging constructs specified in the other child  
 2800 elements can be both produced as well as processed at the software *Message* service layer.

2801 At least one of the *generate* or *parse* attributes MUST be true.

2802

### 2803 **8.6.2 CompositeList element**

2804 The final child element of *Packaging* is *CompositeList*, which is a container for the specific way  
2805 in which the simple parts are combined into groups (MIME multipart) or encapsulated within  
2806 security-related MIME content-types. The *CompositeList* element SHALL be omitted from  
2807 *Packaging* when no security encapsulations or composite multipart are used. When the  
2808 *CompositeList* element is present, the content model for the *CompositeList* element is a  
2809 repeatable sequence of choices of *Composite* or *Encapsulation* elements. The *Composite* and  
2810 *Encapsulation* elements can appear intermixed as desired. The sequence in which the choices  
2811 are presented is important because, given the recursive character of MIME packaging,  
2812 composites or encapsulations can include previously mentioned composites (or rarely,  
2813 encapsulations) in addition to the *Message* parts characterized within the *SimplePart* subtree.  
2814 Therefore, the "top-level" packaging will be described last in the sequence.

2815

2816 The *Composite* element has the following attributes:

- 2817 • a REQUIRED *mimetype* attribute,
- 2818 • a REQUIRED *id* attribute,
- 2819 • an IMPLIED *mimeparameters* attribute.

2820

2821 The *mimetype* attribute provides the value of the MIME content-type for this *Message* part, and  
2822 this will be some MIME composite type, such as "multipart/related" or "multipart/signed". The  
2823 *id* attribute, type ID, provides a way to refer to this composite if it needs to be mentioned as a  
2824 constituent of some later element in the sequence. The *mimeparameters* attribute provides the  
2825 values of any significant MIME parameter (such as "type=application/ xml") that is needed to  
2826 understand the processing demands of the content-type.

2827

2828 The *Composite* element has one child element, *Constituent*.

2829

2830 The *Constituent* element has one REQUIRED attribute, *idref* of type IDREF, an IMPLIED  
2831 boolean attribute *excludeFromSignature*, and two IMPLIED nonNegativeInteger attributes,  
2832 *minOccurs* and *maxOccurs*.

2833

2834 The *idref* attribute has as its value the value of the *id* attribute of a previous *Composite*,  
2835 *Encapsulation*, or *SimplePart* element. The purpose of this sequence of *Constituents* is to  
2836 indicate both the contents and the order of what is packaged within the current *Composite* or  
2837 *Encapsulation*.

2838

2839 The *excludeFromSignature* attribute indicates that this *Constituent* is not to be included as part  
2840 of the ebXML message [XMLDSIG] signature. In other words, the signature generated by the  
2841 *Message Service Handler* should not include a *ds:Reference* element to provide a digest for this  
2842 *Constituent* of the *Message*. This attribute is applicable only if the *Constituent* is part of the top-  
2843 level *Composite* that corresponds to the entire ebXML *Message*.

2844

2845 The *minOccurs* and *maxOccurs* attributes serve to specify the value or range of values that the

2846 referred to item may occur within *Composite*. When unused, it is understood that the item is used  
2847 exactly once.

2848  
2849 The *Encapsulation* element is typically used to indicate the use of MIME security mechanisms,  
2850 such as [S/MIME] or Open-PGP[RFC2015]. A security body part can encapsulate a MIME part  
2851 that has been previously characterized. For convenience, all such security structures are under  
2852 the *Encapsulation* element, even when technically speaking the data is not "inside" the body  
2853 part. (In other words, the so-called clear-signed or detached signature structures possible with  
2854 MIME multipart/signed are for simplicity found under the *Encapsulation* element.)

2855  
2856 Another possible use of the *Encapsulation* element is to represent the application of a  
2857 compression algorithm such as gzip [ZLIB] to some part of the payload, prior to its being  
2858 encrypted and or signed.

2859  
2860 The *Encapsulation* element has the following attributes:

- 2861 • a REQUIRED *mimetype* attribute,
- 2862 • a REQUIRED *id* attribute,
- 2863 • an IMPLIED *mimeparameters* attribute.

2864  
2865 The *mimetype* attribute provides the value of the MIME content-type for this *Message* part, such  
2866 as "application/pkcs7-mime". The *id* attribute, type ID, provides a way to refer to this  
2867 encapsulation if it needs to be mentioned as a constituent of some later element in the sequence.  
2868 The *mimeparameters* attribute provides the values of any significant MIME parameter(s)  
2869 needed to understand the processing demands of the content-type.

2870  
2871 Both the *Encapsulation* element and the *Composite* element have child elements consisting of a  
2872 *Constituent* element or of a repeatable sequence of *Constituent* elements, respectively.

2873  
2874 The *Constituent* element also has zero or one *SignatureTransform* child element and zero or  
2875 one *EncryptionTransform* child element. The *SignatureTransform* element is intended for use  
2876 with XML Digital Signature [XMLDSIG]. When present, it identifies the transforms that must  
2877 be applied to the source data before a digest is computed. The *EncryptionTransform* element is  
2878 intended for use with XML Encryption [XMLENC]. When present, it identifies the transforms  
2879 that must be applied to a *CipherReference* before decryption can be performed. The  
2880 *SignatureTransforms* element and the *EncryptionTransforms* element each contains one or  
2881 more *ds:Transform* [XMLDSIG] elements.

## 2882 2883 **8.7 Signature element**

2884 The *Signature* element (cardinality zero or one) enables the CPA to be digitally signed using  
2885 technology that conforms with the XML Digital Signature specification[XMLDSIG]. The  
2886 *Signature* element is the root of a subtree of elements used for signing the *CPP*. The syntax is:

```
2887 <tp:Signature>...</tp:Signature>
```

2888  
2889 The *Signature* element contains one or more *ds:Signature* elements. The content of the

2891 ***ds:Signature*** element and any subelements are defined by the XML Digital Signature  
2892 specification. See Section 9.9 for a detailed discussion.

2893  
2894 NOTE: It is necessary to wrap the ***ds:Signature*** elements with a **Signature** element in the  
2895 target namespace to allow for the possibility of having wildcard elements (with  
2896 namespace="##other") within the **CollaborationProtocolProfile** and  
2897 **CollaborationProtocolAgreement** elements. The content model would be ambiguous without  
2898 the wrapping.

2899  
2900 The following additional constraints on ***ds:Signature*** are imposed:

- 2901
- 2902 • A *CPP* MUST be considered invalid if any ***ds:Signature*** element fails core validation as  
2903 defined by the XML Digital Signature specification[XMLDSIG].
- 2904
- 2905 • Whenever a *CPP* is signed, each ***ds:Reference*** element within a ***ProcessSpecification***  
2906 element MUST pass reference validation and each ***ds:Signature*** element MUST pass  
2907 core validation.
- 2908

2909 NOTE: In case a *CPP* is unsigned, software might nonetheless validate the ***ds:Reference***  
2910 elements within ***ProcessSpecification*** elements and report any exceptions.

2911  
2912 NOTE: Software for creation of *CPPs* and *CPAs* MAY recognize ***ds:Signature*** and  
2913 automatically insert the element structure necessary to define signing of the *CPP* and *CPA*.  
2914 Signature generation is outlined in Section 9.9.1.1; details of the cryptographic process are  
2915 outside the scope of this specification.

2916  
2917 NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which validity  
2918 tests MAY be made.

2919

## 2920 8.8 Comment element

2921 The ***CollaborationProtocolProfile*** element contains zero or more ***Comment*** elements. The  
2922 ***Comment*** element is a textual note that can be added to serve any purpose the author desires.  
2923 The language of the ***Comment*** is identified by a REQUIRED ***xml:lang*** attribute. The ***xml:lang***  
2924 attribute MUST comply with the rules for identifying languages specified in [XML]. If multiple  
2925 ***Comment*** elements are present, each can have a different ***xml:lang*** attribute value. An example  
2926 of a ***Comment*** element follows:

2927  
2928 

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

2929  
2930 When a *CPA* is composed from two *CPPs*, all ***Comment*** elements from both *CPPs* SHALL be  
2931 included in the *CPA* unless the two *Parties* agree otherwise.

## 2932 9 CPA Definition

2933 A *Collaboration-Protocol Agreement (CPA)* defines the capabilities that two *Parties* need to  
 2934 agree upon to enable them to engage in electronic *Business* for the purposes of the particular  
 2935 *CPA*. This section defines and discusses the details of the *CPA*. The discussion is illustrated with  
 2936 some XML fragments.

2937  
 2938 Most of the XML elements in this section are described in detail in Section 8, "CPP Definition".  
 2939 In general, this section does not repeat that information. The discussions in this section are  
 2940 limited to those elements that are not in the *CPP* or for which additional discussion is needed in  
 2941 the *CPA* context. See also Appendix D for the XML Schema, and Appendix B for an example of  
 2942 a *CPA* document.

2943

### 2944 9.1 CPA Structure

2945 Following is the overall structure of the *CPA*:

2946

```

2947     <CollaborationProtocolAgreement
2948         xmlns:tp="http://www.oasis-open.org/committees/ebxml-
2949 cppa/schema/cpp-cpa-2_0.xsd"
2950         xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2951         xmlns:xlink="http://www.w3.org/1999/xlink"
2952         tp:cpaid="YoursAndMyCPA"
2953         tp:version="2.0a">
2954     <tp:Status tp:value="proposed"/>
2955     <tp:Start>1988-04-07T18:39:09</Start>
2956     <tp:End>1990-04-07T18:40:00</End>
2957     <!-- ConversationConstraints MAY appear 0 or 1 times -->
2958     <tp:ConversationConstraints
2959         tp:invocationLimit="100"
2960         tp:concurrentConversations="4"/>
2961     <tp:PartyInfo>
2962         ...
2963     </tp:PartyInfo>
2964     <tp:PartyInfo>
2965         ...
2966     </tp:PartyInfo>
2967     <tp:SimplePart> <!-- one or more -->
2968         ...
2969     </tp:SimplePart>
2970     <tp:Packaging tp:id="N20"> <!-- one or more -->
2971         ...
2972     </tp:Packaging>
2973     <tp:Signature> <!-- zero or one time -->
2974         ...
2975     </tp:Signature>
2976     <tp:Comment xml:lang="en-GB">any text</Comment> <!-- zero or more -
2977 -->
2978     </tp:CollaborationProtocolAgreement>
2979
  
```

## 2980 9.2 CollaborationProtocolAgreement element

2981 The *CollaborationProtocolAgreement* element is the root element of a *CPA*. It has a  
2982 REQUIRED *cpaid* attribute that supplies a unique identifier for the document. The value of the  
2983 *cpaid* attribute SHALL be assigned by one *Party* and used by both. It is RECOMMENDED that  
2984 the value of the *cpaid* attribute be a URI. The value of the *cpaid* attribute SHALL be used as the  
2985 value of the *CPAId* element in the ebXML *Message Header*[ebMS] or of a similar element in a  
2986 *Message Header* of an alternative messaging service.

2987

2988 NOTE: Each *Party* might associate a local identifier with the *cpaid* attribute.

2989

2990 In addition, the *CollaborationProtocolAgreement* element has a REQUIRED *version* attribute.  
2991 This attribute indicates the version of the schema to which the *CPA* conforms. The value of the  
2992 *version* attribute SHOULD be a string such as "2\_0a" or "2\_0b". The value of the *cpaid* attribute  
2993 SHOULD be changed with each change made to the *CPA* document both during negotiation and  
2994 after it has been deployed.

2995

2996 NOTE: The method of assigning unique *cpaid* values is left to the implementation.

2997

2998 The *CollaborationProtocolAgreement* element has REQUIRED [XML] Namespace[XMLNS]  
2999 declarations that are defined in Section 8, "CPP Definition".

3000

3001 The *CollaborationProtocolAgreement* element is comprised of the following child elements,  
3002 most of which are described in greater detail in subsequent sections:

- 3003 • a REQUIRED *Status* element that identifies the state of the process that creates the  
3004 *CPA*,
- 3005 • a REQUIRED *Start* element that records the date and time that the *CPA* goes into  
3006 effect,
- 3007 • a REQUIRED *End* element that records the date and time after which the *CPA*  
3008 MUST be renegotiated by the *Parties*,
- 3009 • zero or one *ConversationConstraints* element that documents certain agreements  
3010 about conversation processing,
- 3011 • two REQUIRED *PartyInfo* elements, one for each *Party* to the *CPA*,
- 3012 • one or more *SimplePart* elements,
- 3013 • one or more *Packaging* elements,
- 3014 • zero or one Signature element that provides for signing of the *CPA* using the XML  
3015 Digital Signature[XMLDSIG] standard,
- 3016 • zero or more *Comment* elements.

3017

## 3018 9.3 Status Element

3019 The *Status* element records the state of the composition/negotiation process that creates the *CPA*.  
3020 An example of the *Status* element follows:

3021

```
3022 <tp:Status tp:value="proposed"/>
```

3023

3024 The Status element has a REQUIRED *value* attribute that records the current state of

3025 composition of the *CPA*. This attribute is an enumeration comprised of the following possible  
3026 values:

- 3027 • "proposed", meaning that the *CPA* is still being negotiated by the *Parties*,
- 3028 • "agreed", meaning that the contents of the *CPA* have been agreed to by both *Parties*,
- 3029 • "signed", meaning that the *CPA* has been "signed" by the *Parties*. This "signing"  
3030 takes the form of a digital signature that is described in Section 9.7 below.

3031  
3032 NOTE: The *Status* element MAY be used by a *CPA* composition and negotiation tool to  
3033 assist it in the process of building a *CPA*.  
3034

## 3035 9.4 CPA Lifetime

3036 The lifetime of the *CPA* is given by the *Start* and *End* elements. The syntax is:

```
3037 <tp:Start>1988-04-07T18:39:09Z</tp:Start>  
3038 <tp:End>1990-04-07T18:40:00Z</tp:End>  
3039  
3040
```

### 3041 9.4.1 Start element

3042 The *Start* element specifies the starting date and time of the *CPA*. The *Start* element SHALL be  
3043 a string value that conforms to the content model of a canonical dateTime as defined in the XML  
3044 Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm UTC  
3045 (Coordinated Universal Time) on May 31, 1999, a *Start* element would have the following value:

```
3046 1999-05-31T13:20:00Z  
3047  
3048
```

3049 The *Start* element SHALL be represented as Coordinated Universal Time (UTC).  
3050

### 3051 9.4.2 End element

3052 The *End* element specifies the ending date and time of the *CPA*. The *End* element SHALL be a  
3053 string value that conforms to the content model of a canonical dateTime as defined in the XML  
3054 Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm UTC  
3055 (Coordinated Universal Time) on May 31, 1999, an *End* element would have the following  
3056 value:

```
3057 1999-05-31T13:20:00Z  
3058  
3059
```

3060 The *End* element SHALL be represented as Coordinated Universal Time (UTC).  
3061

3062 When the end of the *CPA*'s lifetime is reached, any *Business Transactions* that are still in  
3063 progress SHALL be allowed to complete and no new *Business Transactions* SHALL be started.  
3064 When all in-progress *Business Transactions* on each conversation are completed, the  
3065 *Conversation* SHALL be terminated whether or not it was completed.  
3066

3067 When a *CPA* is signed, software for signing the agreements SHALL warn if any signing  
3068 certificate's validity expires prior to the proposed time for ending the *CPA*. The opportunity to



3069 renegotiate a *CPA End* value or to in some other way align certificate validity periods with CPA  
 3070 validity periods SHALL be made available. (Other ways to align these validity periods would  
 3071 include reissuing the signing certificates for a longer period or obtaining new certificates for this  
 3072 purpose.)

3073  
 3074 Signing software SHOULD also attempt to align the validity periods of certificates referred to  
 3075 within the CPA that perform security functions so as to not expire before the CPA expires. This  
 3076 alignment can occur in several ways including making use of *ds:KeyInfo*'s content model  
 3077 *ds:RetrievalMethod* so that a new certificate can be installed and still be retrieved in accordance  
 3078 with the information in *ds:RetrievalMethod*. If no alignment can be attained, signing software  
 3079 MUST warn the user of the situation that the *CPA* validity exceeds the validity of some of the  
 3080 certificates referred to within the *CPA*.

3081  
 3082 NOTE: If a *Business* application defines a conversation as consisting of multiple *Business*  
 3083 *Transactions*, such a conversation MAY be terminated with no error indication when the  
 3084 end of the lifetime is reached. The run-time system could provide an error indication to  
 3085 the application.

3086  
 3087 NOTE: It might not be feasible to wait for outstanding conversations to terminate before  
 3088 ending the *CPA* since there is no limit on how long a conversation can last.

3089  
 3090 NOTE: The run-time system SHOULD return an error indication to both *Parties* when a  
 3091 new *Business Transaction* is started under this *CPA* after the date and time specified in  
 3092 the *End* element.

3093

## 3094 9.5 ConversationConstraints Element

3095 The *ConversationConstraints* element places limits on the number of conversations under the  
 3096 *CPA*. An example of this element follows:

```
3097
3098 <tp:ConversationConstraints tp:invocationLimit="100"
3099   tp:concurrentConversations="4"/>
```

3100

3101 The *ConversationConstraints* element has the following attributes:

- 3102 • an IMPLIED *invocationLimit* attribute,
- 3103 • an IMPLIED *concurrentConversations* attribute.

3104

### 3105 9.5.1 invocationLimit attribute

3106 The *invocationLimit* attribute defines the maximum number of conversations that can be  
 3107 processed under the *CPA*. When this number has been reached, the *CPA* is terminated and  
 3108 MUST be renegotiated. If no value is specified, there is no upper limit on the number of  
 3109 conversations and the lifetime of the *CPA* is controlled solely by the *End* element.

3110

3111 NOTE: The *invocationLimit* attribute sets a limit on the number of units of *Business* that  
 3112 can be performed under the *CPA*. It is a *Business* parameter, not a performance  
 3113 parameter.

3114

### 3115 **9.5.2 concurrentConversations attribute**

3116 The *concurrentConversations* attribute defines the maximum number of conversations that can  
3117 be in process under this *CPA* at the same time. If no value is specified, processing of concurrent  
3118 conversations is strictly a local matter.

3119

3120 NOTE: The *concurrentConversations* attribute provides a parameter for the *Parties* to use  
3121 when it is necessary to limit the number of conversations that can be concurrently processed  
3122 under a particular *CPA*. For example, the back-end process might only support a limited  
3123 number of concurrent conversations. If a request for a new conversation is received when  
3124 the maximum number of conversations allowed under this *CPA* is already in process, an  
3125 implementation MAY reject the new conversation or MAY enqueue the request until an  
3126 existing conversation ends. If no value is given for *concurrentConversations*, how to handle  
3127 a request for a new conversation for which there is no capacity is a local implementation  
3128 matter.

3129

## 3130 **9.6 PartyInfo Element**

3131 The general characteristics of the *PartyInfo* element are discussed in Section 8.4.

3132

3133 The *CPA* SHALL have one *PartyInfo* element for each *Party* to the *CPA*. The *PartyInfo*  
3134 element specifies the *Parties'* agreed terms for engaging in the *Business Collaborations* defined  
3135 by the *Process-Specification* documents referenced by the *CPA*. If a *CPP* has more than one  
3136 *PartyInfo* element, the appropriate *PartyInfo* element SHALL be selected from each *CPP* when  
3137 composing a *CPA*.

3138

3139 In the *CPA*, there SHALL be one or more *PartyId* elements under each *PartyInfo* element. The  
3140 values of these elements are the same as the values of the *PartyId* elements in the ebXML  
3141 *Message Service* specification[ebMS] or similar messaging service specification. These *PartyId*  
3142 elements SHALL be used within a *To* or *From Header* element of an ebXML *Message*.

3143

### 3144 **9.6.1 ProcessSpecification element**

3145 The *ProcessSpecification* element identifies the *Business Collaboration* that the two *Parties*  
3146 have agreed to perform. There can be one or more *ProcessSpecification* elements in a *CPA*.  
3147 Each SHALL be a child element of a separate *CollaborationRole* element. See the discussion in  
3148 Section 8.4.3.

3149

## 3150 **9.7 SimplePart element**

3151 The *CollaborationProtocolAgreement* element SHALL contain one or more *SimplePart*  
3152 elements. See Section 8.5 for details of the syntax of the *SimplePart* element.

## 3153 **9.8 Packaging element**

3154 The *CollaborationProtocolAgreement* element SHALL contain one or more *Packaging*  
3155 elements. See Section 8.6 for details of the syntax of the *Packaging* element.

3156

## 3157 9.9 Signature element

3158 A *CPA* document can be digitally signed by one or more of the *Parties* as a means of ensuring its  
3159 integrity as well as a means of expressing the agreement just as a corporate officer's signature  
3160 would do for a paper document. If signatures are being used to digitally sign an ebXML *CPA* or  
3161 *CPP* document, then [XMLDSIG] SHALL be used to digitally sign the document.

3162

3163 The *Signature* element, if present, is made up of one or more *ds:Signature* elements. In a *CPA*  
3164 involving two *Parties*, there can be up to three *ds:Signature* elements within the *Signature*  
3165 element. The *CPA* is initially signed by one of the two *Parties*. The other *Party* could then sign  
3166 over the first *Party's* signature. The resulting *CPA* MAY then be signed by a notary.

3167

3168 The *ds:Signature* element is the root of a subtree of elements used for signing the *CPP*.

3169

3170 The content of this element and any subelements are defined by the XML Digital Signature  
3171 specification[XMLDSIG]. The following additional constraints on *ds:Signature* are imposed:

3172

- 3173 • A *CPA* MUST be considered invalid if any *ds:Signature* fails core validation as defined  
3174 by the XML Digital Signature specification.

3175

- 3176 • Whenever a *CPA* is signed, each *ds:Reference* within a *ProcessSpecification* MUST  
3177 pass reference validation and each *ds:Signature* MUST pass core validation.

3178

3179 NOTE: In case a *CPA* is unsigned, software MAY nonetheless validate the *ds:Reference*  
3180 elements within *ProcessSpecification* elements and report any exceptions.

3181

3182 Software for creation of *CPPs* and *CPAs* SHALL recognize *ds:Signature* and automatically  
3183 insert the element structure necessary to define signing of the *CPP* and *CPA*. Signature creation  
3184 itself is a cryptographic process that is outside the scope of this specification.

3185

3186 NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which a *CPA*  
3187 MAY be validated.

3188

### 3189 9.9.1 Persistent Digital Signature

3190 If [XMLDSIG] is used to sign an ebXML *CPP* or *CPA*, the process defined in this section of the  
3191 specification SHALL be used.

3192

#### 3193 9.9.1.1 Signature Generation

3194 Following are the steps to create a digital signature:

- 3195 1. Create a *SignedInfo* element, a child element of *ds:Signature*. *SignedInfo* SHALL have  
3196 child elements *SignatureMethod*, *CanonicalizationMethod*, and *Reference* as prescribed by  
3197 [XMLDSIG].

- 3198 2. Canonicalize and then calculate the *SignatureValue* over *SignedInfo* based on algorithms  
3199 specified in *SignedInfo* as specified in [XMLDSIG].

- 3200 3. Construct the *Signature* element that includes the *SignedInfo*, *KeyInfo*  
 3201 (RECOMMENDED), and *SignatureValue* elements as specified in [XMLDSIG].  
 3202 4. Include the namespace qualified *Signature* element in the document just signed, following  
 3203 the last *PartyInfo* element.

3204

### 9.9.1.2 ds:SignedInfo element

3205 The *ds:SignedInfo* element SHALL be comprised of zero or one *ds:CanonicalizationMethod*  
 3206 element, the *ds:SignatureMethod* element, and one or more *ds:Reference* elements.  
 3207

3208

### 9.9.1.3 ds:CanonicalizationMethod element

3209 The *ds:CanonicalizationMethod* element as defined in [XMLDSIG], can occur zero or one  
 3210 time, meaning that the element need not appear in an instance of a *ds:SignedInfo* element. The  
 3211 default canonicalization method that is applied to the data to be signed is [XMLC14N] in the  
 3212 absence of a *ds:CanonicalizationMethod* element that specifies otherwise. This default SHALL  
 3213 also serve as the default canonicalization method for the ebXML *CPP* and *CPA* documents.  
 3214

3215

### 9.9.1.4 ds:SignatureMethod element

3216 The *ds:SignatureMethod* element SHALL be present and SHALL have an *Algorithm* attribute.  
 3217 The RECOMMENDED value for the *Algorithm* attribute is:  
 3218

3219

3220 "http://www.w3.org/2000/09/xmlsig#sha1"

3221

3222 This RECOMMENDED value SHALL be supported by all compliant ebXML *CPP* or *CPA*  
 3223 software implementations.

3224

### 9.9.1.5 ds:Reference element

3225 The *ds:Reference* element for the *CPP* or *CPA* document SHALL have a REQUIRED URI  
 3226 attribute value of "" to provide for the signature to be applied to the document that contains the  
 3227 *ds:Signature* element (the *CPA* or *CPP* document). The *ds:Reference* element for the *CPP* or  
 3228 *CPA* document can include an IMPLIED *type* attribute that has a value of:  
 3229

3230

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

3232

3233 in accordance with [XMLDSIG]. This attribute is purely informative. It MAY be omitted.  
 3234 Implementations of software designed to author or process an ebXML *CPA* or *CPP* document  
 3235 SHALL be prepared to handle either case. The *ds:Reference* element can include the *id* attribute,  
 3236 type ID, by which this *ds:Reference* element is referenced from a *ds:Signature* element.  
 3237

3238

### 9.9.1.6 ds:Transform element

3239 The *ds:Reference* element for the *CPA* or *CPP* document SHALL include a descendant  
 3240 *ds:Transform* element that excludes the containing *ds:Signature* element and all its descendants.  
 3241 This exclusion is achieved by means of specifying the *ds:Algorithm* attribute of the *Transform*  
 3242 element as

3243

3244 "http://www.w3.org/2000/09/xmlsig#enveloped-signature"

3245

3246 For example:

<ds:Reference ds:URI="">

```

3247     <ds:Transforms>
3248         <ds:Transform
3249             ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
3250     </ds:Transforms>
3251     <ds:DigestMethod
3252         ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3253     <ds:DigestValue>...</ds:DigestValue>
3254 </ds:Reference>
3255

```

### 3256 9.9.1.7 ds:Algorithm attribute

3257 The *ds:Transform* element SHALL include a *ds:Algorithm* attribute that has a value of:  
3258 `http://www.w3.org/2000/09/xmldsig#enveloped-signature`

3259

3260 NOTE: When digitally signing a *CPA*, it is RECOMMENDED that each *Party* sign the  
3261 document in accordance with the process described above.

3262

3263 When the two Parties sign the *CPA*, the first *Party* that signs the *CPA* SHALL sign only the *CPA*  
3264 contents, excluding their own signature. The second *Party* SHALL sign over the contents of the  
3265 *CPA* as well as the *ds:Signature* element that contains the first *Party's* signature. If necessary, a  
3266 notary can then sign over both signatures.

3267

## 3268 9.10 Comment element

3269 The *CollaborationProtocolAgreement* element contains zero or more *Comment* elements. See  
3270 Section 8.8 for details of the syntax of the *Comment* element.

3271

## 3272 9.11 Composing a CPA from Two CPPs

3273 This section discusses normative issues in composing a *CPA* from two *CPPs*. See also Appendix  
3274 E, "CPA Composition (Non-Normative)".

3275

### 3276 9.11.1 ID Attribute Duplication

3277 In composing a *CPA* from two *CPPs*, there is a hazard that ID attributes from the two *CPPs*  
3278 might have duplicate values. When a *CPA* is composed from two *CPPs*, duplicate ID attribute  
3279 values SHALL be tested for. If a duplicate ID attribute value is present, one of the duplicates  
3280 SHALL be given a new value and the corresponding IDREF attribute values from the  
3281 corresponding *CPP* SHALL be corrected.

3282

3283 NOTE: A party can seek to prevent ID/IDREF reassignment in the *CPA* by choosing ID  
3284 and IDREF values which are likely to be unique among its trading partners. For example,  
3285 the following *Certificate* element found in a *CPP* has a *certId* attribute that is generic  
3286 enough that it might clash with a *certId* attribute found in a collaborating party's *CPP*:

```

3287 <tp:Certificate
3288     tp:certId="EncryptionCert"><ds:KeyInfo/></tp:Certificate>
3289

```

3290 To prevent reassignment of this ID (and its associated IDREFs) in a *CPA*, a better choice  
3291 of *certId* in Company A's *CPP* would be:

3292  
3293           <tp:Certificate  
3294           tp:certId="CompanyA\_EncryptionCert"><ds:KeyInfo/></tp:Certificate>  
3295

## 3296 **9.12 Modifying Parameters of the Process-Specification Document Based on** 3297 **Information in the CPA**

3298   A *Process-Specification* document contains a number of parameters, expressed as XML  
3299   attributes. An example is the security attributes that are counterparts of the attributes of the *CPA*  
3300   *BusinessTransactionCharacteristics* element. The values of these attributes can be considered to  
3301   be default values or recommendations. When a *CPA* is created, the *Parties* might decide to  
3302   accept the recommendations in the *Process-Specification* or they *MAY* agree on values of these  
3303   parameters that better reflect their needs.

3304  
3305   When a *CPA* is used to configure a run-time system, choices specified in the *CPA* *MUST* always  
3306   assume precedence over choices specified in the referenced *Process-Specification* document. In  
3307   particular, all choices expressed in a *CPA*'s *BusinessTransactionCharacteristics* and *Packaging*  
3308   elements *MUST* be implemented as agreed to by the *Parties*. These choices *SHALL* override  
3309   the default values expressed in the *Process-Specification* document. The process of installing the  
3310   information from the *CPA* and *Process-Specification* document *MUST* verify that all of the  
3311   resulting choices are mutually consistent and *MUST* signal an error if they are not.

3312  
3313       NOTE: There are several ways of overriding the information in the *Process-*  
3314       *Specification* document by information from the *CPA*. For example:

- 3315
- 3316       • The *CPA* composition tool can create a separate copy of the *Process-Specification*  
3317       document. The tool can then directly modify the *Process-Specification* document  
3318       with information from the *CPA*. An advantage of this method is that the override  
3319       process is performed entirely by the *CPA* composition tool.
  - 3320       • A *CPA* installation tool can dynamically override parameters in the *Process-*  
3321       *Specification* document using information from the corresponding parameters in the  
3322       *CPA* at the time the *CPA* and *Process-Specification* document are installed in the  
3323       *Parties'* systems. This eliminates the need to create a separate copy of the *Process-*  
3324       *Specification* document.
  - 3325       • Other possible methods might be based on XSLT transformations of the parameter  
3326       information in the *CPA* and/or the *Process-Specification* document.

3327 

## 10 References

3328 Some references listed below specify functions for which specific XML definitions are provided  
3329 in the *CPP* and *CPA*. Other specifications are referred to in this specification in the sense that  
3330 they are represented by keywords for which the *Parties* to the *CPA* MAY obtain plug-ins or  
3331 write custom support software but do not require specific XML element sets in the *CPP* and  
3332 *CPA*.

3333  
3334 In a few cases, the only available specification for a function is a proprietary specification.  
3335 These are indicated by notes within the citations below.

3336  
3337 [ccOVER] ebXML Core Components Overview, <http://www.ebxml.org/specs/ccOVER.pdf>.

3338  
3339 [DIGENV] Digital Envelope, RSA Laboratories, <http://www.rsasecurity.com/rsalabs/faq/2-2-4.html>.  
3340 NOTE: At this time, the only available specification for digital envelope appears to be the RSA  
3341 Laboratories specification.

3342  
3343 [ebBPSS] ebXML Business Process Specification Schema, <http://www.ebxml.org/specs/ebBPSS.pdf>.

3344  
3345 [ebMS] ebXML Message Service Specification, [http://www.oasis-open.org/committees/ebxml-](http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf)  
3346 [msg/documents/ebMS\\_v2\\_0.pdf](http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf).

3347  
3348 [ebRS] ebXML Registry Services Specification, [http://www.oasis-](http://www.oasis-open.org/committees/regist/documents/2.0/specs/ebrs.pdf)  
3349 [open.org/committees/regist/documents/2.0/specs/ebrs.pdf](http://www.oasis-open.org/committees/regist/documents/2.0/specs/ebrs.pdf).

3350  
3351 [HTTP] Hypertext Transfer Protocol, Internet Engineering Task Force RFC 2616, [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc2616.txt)  
3352 [editor.org/rfc/rfc2616.txt](http://www.rfc-editor.org/rfc/rfc2616.txt).

3353  
3354 [IPSEC] IP Security Document Roadmap, Internet Engineering Task Force RFC 2411,  
3355 <http://www.ietf.org/rfc/rfc2411.txt>.

3356  
3357 [ISO6523] Structure for the Identification of Organizations and Organization Parts, International  
3358 Standards Organization ISO-6523.

3359  
3360 [MIME] MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying  
3361 and Describing the Format of Internet *Message* Bodies. Internet Engineering Task Force RFC  
3362 1521, <http://www.ietf.org/rfc/rfc1521.txt>.

3363  
3364 [RFC959] File Transfer Protocol (FTP), Internet Engineering Task Force RFC 959,  
3365 <http://www.ietf.org/rfc/rfc959.txt>.

3366  
3367 [RFC1123] Requirements for Internet Hosts -- Application and Support, R. Braden, Internet  
3368 Engineering Task Force, October 1989, <http://www.ietf.org/rfc/rfc1123.txt>.

3369  
3370 [RFC1579] Firewall-Friendly FTP, S. Bellovin, Internet Engineering Task Force, February 1994,

- 3371 <http://www.ietf.org/rfc/rfc1579.txt> .
- 3372
- 3373 [RFC2015] MIME Security with Pretty Good Privacy, M. Elkins, Internet Engineering Task
- 3374 Force, RFC 2015, <http://www.ietf.org/rfc/rfc2015.txt> .
- 3375
- 3376 [RFC2119] Key Words for use in RFCs to Indicate Requirement Levels, Internet Engineering
- 3377 Task Force RFC 2119, <http://www.ietf.org/rfc/rfc2119.txt> .
- 3378
- 3379 [RFC2246] The TLS Protocol, <http://www.ietf.org/rfc/rfc2246.txt> .
- 3380
- 3381 [RFC2251] Lightweight Directory Access Protocol (v3); Mark Wahl, Tim Howes, Steve Kille,
- 3382 <http://www.ietf.org/rfc/rfc2251.txt> .
- 3383
- 3384 [RFC2396] Uniform Resource Identifiers (URI): Generic Syntax; T. Berners-Lee, R. Fielding, L.
- 3385 Masinter - August 1998, <http://www.ietf.org/rfc/rfc2396.txt> .
- 3386
- 3387 [RFC2617] HTTP Authentication: Basic and Digest Authentication, J. Franks, P. Hallam-Baker,
- 3388 J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, Internet Engineering Task Force,
- 3389 June 1999, <http://www.ietf.org/rfc/rfc2617.txt> .
- 3390
- 3391 [RFC2822] Internet Message Format, Internet Engineering Task Force RFC 2822,
- 3392 <http://www.ietf.org/rfc/rfc2822.txt> .
- 3393
- 3394 [S/MIME] S/MIME Version 3 Message Specification, Internet Engineering Task Force RFC
- 3395 2633, <http://www.ietf.org/rfc/rfc2633.txt> .
- 3396
- 3397 [SAML] Security Assertion Markup Language, <http://www.oasis-open.org/committees/security/->
- 3398 [documents](http://www.oasis-open.org/committees/security/-) .
- 3399
- 3400 [SMTP] Simple Mail Transfer Protocol, Internet Engineering Task Force RFC 821,
- 3401 <http://www.faqs.org/rfcs/rfc821.html> .
- 3402
- 3403 [SSL] Secure Sockets Layer, Netscape Communications Corp., <http://www.netscape.com/eng/ssl3/>
- 3404 NOTE: At this time, it appears that the Netscape specification is the only available specification
- 3405 of SSL.
- 3406
- 3407 [X12] ANSI X12 Standard for Electronic Data Interchange, X12 Standard Release
- 3408 4050, December 2001.
- 3409
- 3410 [XAML] Transaction Authority Markup Language, <http://xaml.org/>.
- 3411
- 3412 [XLINK] XML Linking Language, <http://www.w3.org/TR/xlink/>.
- 3413
- 3414 [XML] Extensible Markup Language (XML), World Wide Web Consortium,
- 3415 <http://www.w3.org/XML> .
- 3416



- 3417 [XMLC14N] Canonical XML, Ver. 1.0, <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>.  
3418  
3419 [XMLDSIG] XML Signature Syntax and Processing, Worldwide Web Consortium,  
3420 <http://www.w3.org/TR/xmlsig-core/>.  
3421  
3422 [XMLENC] XML Encryption Syntax and Processing, W3C Candidate Recommendation 04  
3423 March 2002, <http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/>.  
3424  
3425 [XMLNS] Namespaces in XML, T. Bray, D. Hollander, and A. Layman, Jan. 1999,  
3426 <http://www.w3.org/TR/REC-xml-names/>.  
3427  
3428 [XMLSCHEMA-1] XML Schema Part 1: Structures, <http://www.w3.org/TR/xmlschema-1/>.  
3429  
3430 [XMLSCHEMA-2] XML Schema Part 2: Datatypes,  
3431 <http://www.w3.org/TR/xmlschema-2/>.  
3432  
3433 [XPOINTER] XML Pointer Language, ver. 1.0, <http://www.w3.org/TR/xptr/>.  
3434  
3435 [ZLIB] Zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library,  
3436 <http://www.gzip.org/zlib/>.

## 3437 11 Conformance

3438 In order to conform to this specification, an implementation:

- 3439 a) SHALL support all the functional and interface requirements defined in this specification,
- 3440 b) SHALL NOT specify any requirements that would contradict or cause non-conformance
- 3441 to this specification.

3442

3443 A conforming implementation SHALL satisfy the conformance requirements of the applicable  
3444 parts of this specification.

3445

3446 An implementation of a tool or service that creates or maintains ebXML *CPP* or *CPA* instance  
3447 documents SHALL be determined to be conformant by validation of the *CPP* or *CPA* instance  
3448 documents, created or modified by said tool or service, against the XML  
3449 Schema[XMLSCHEMA-1] definition of the *CPP* or *CPA* in Appendix D and available from

3450

3451 [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)

3452

3453 by using two or more validating XML Schema parsers that conform to the W3C XML Schema  
3454 specifications[XMLSCHEMA-1, XMLSCHEMA-2].

3455

3456 The objective of conformance testing is to determine whether an implementation being tested  
3457 conforms to the requirements stated in this specification. Conformance testing enables vendors to  
3458 implement compatible and interoperable systems. Implementations and applications SHALL be  
3459 tested using available test suites to verify their conformance to this specification.

3460

3461 Publicly available test suites from vendor neutral organizations such as OASIS and the U.S.A.  
3462 National Institute of Science and Technology (NIST) SHOULD be used to verify the  
3463 conformance of implementations, applications, and components claiming conformance to this  
3464 specification. Open-source reference implementations might be available to allow vendors to test  
3465 their products for interface compatibility, conformance, and interoperability.

3466

## 3467 12 Disclaimer

3468 The views and specification expressed in this document are those of the authors and are not  
3469 necessarily those of their employers. The authors and their employers specifically disclaim  
3470 responsibility for any problems arising from correct or incorrect implementation or use of this  
3471 design.

## 3472 13 Contact Information

3473

3474 Arvola Chan (Author)

3475 TIBCO Software

3476 3165 Porter Drive

3477 Palo Alto, CA 94304

3478 USA

3479 Phone: 650-846-5046

3480 email: <mailto:arvola@tibco.com>3481 <mailto:>

3482 Dale W. Moberg (Author)

3483 Cyclone Commerce

3484 8388 E. Hartford Drive

3485 Scottsdale, AZ 85255

3486 USA

3487 Phone: 480-627-2648

3488 email: <mailto:dmoberg@cyclonecommerce.com>

3489

3490 Himagiri Mukkamala (Author)

3491 Sybase Inc.

3492 5000 Hacienda Dr

3493 Dublin, CA, 84568

3494 USA

3495 Phone: 925-236-5477

3496 email: <mailto:himagiri@sybase.com>

3497

3498 Peter M. Ogden (Author)

3499 Cyclone Commerce, Inc.

3500 8388 East Hartford Drive

3501 Scottsdale, AZ 85255

3502 USA

3503 Phone: 480-627-1800

3504 email: <mailto:pogden@cyclonecommerce.com>

3505

3506 Martin W. Sachs (Author)

3507 IBM T. J. Watson Research Center

3508 P.O.B. 704

3509 Yorktown Hts, NY 10598

3510 USA

3511 Phone: 914-784-7287

3512 email: <mailto:mwsachs@us.ibm.com>

3513

3514 Tony Weida (Coordinating Editor)

3515 535 West 110<sup>th</sup> St., #4J

3516 New York, NY 10025

3517 USA  
3518 Phone: 212-678-5265  
3519 email: <mailto:rweida@hotmail.com>  
3520  
3521 Jean Zheng  
3522 Vitria  
3523 945 Stewart Drive  
3524 Sunnyvale, CA 94086  
3525 USA  
3526 Phone: 408-212-2468  
3527 email: <mailto:jzheng@vitria.com>

## 3528 Notices

3529

3530 OASIS takes no position regarding the validity or scope of any intellectual property or other  
3531 rights that might be claimed to pertain to the implementation or use of the technology described  
3532 in this document or the extent to which any license under such rights might or might not be  
3533 available; neither does it represent that it has made any effort to identify any such rights.

3534 Information on OASIS's procedures with respect to rights in OASIS specifications can be found  
3535 at the OASIS website. Copies of claims of rights made available for publication and any  
3536 assurances of licenses to be made available, or the result of an attempt made to obtain a general  
3537 license or permission for the use of such proprietary rights by implementors or users of this  
3538 specification, can be obtained from the OASIS Executive Director.

3539

3540 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
3541 applications, or other proprietary rights which may cover technology that may be required to  
3542 implement this specification. Please address the information to the OASIS Executive Director.

3543

3544 Copyright (C) The Organization for the Advancement of Structured Information Standards  
3545 [OASIS] 2001. All Rights Reserved.

3546

3547 This document and translations of it may be copied and furnished to others, and derivative works  
3548 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
3549 published and distributed, in whole or in part, without restriction of any kind, provided that the  
3550 above copyright notice and this paragraph are included on all such copies and derivative works.  
3551 However, this document itself may not be modified in any way, such as by removing the  
3552 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
3553 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
3554 Property Rights document must be followed, or as required to translate it into languages other  
3555 than English.

3556

3557 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
3558 successors or assigns.

3559

3560 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
3561 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT  
3562 LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN  
3563 WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF  
3564 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

3565

## 3566 Copyright Statement

3567  
3568 Copyright © UN/CEFACT and OASIS, 2001. All Rights Reserved.

3569  
3570 This document and translations of it MAY be copied and furnished to others, and derivative  
3571 works that comment on or otherwise explain it or assist in its implementation MAY be prepared,  
3572 copied, published and distributed, in whole or in part, without restriction of any kind, provided  
3573 that the above copyright notice and this paragraph are included on all such copies and derivative  
3574 works. However, this document itself MAY not be modified in any way, such as by removing  
3575 the copyright notice or references to ebXML, UN/CEFACT, or OASIS, except as required to  
3576 translate it into languages other than English.

3577  
3578 The limited permissions granted above are perpetual and will not be revoked by ebXML or its  
3579 successors or assigns.

3580  
3581 This document and the information contained herein is provided on an "AS IS" basis and  
3582 ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT  
3583 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN  
3584 WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF  
3585 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

3586

## 3587 Appendix A Example of CPP Document (Non-Normative)

3588 This example includes two CPPs that are used to form the CPA in Appendix B. They are  
3589 available as ASCII files at

3590 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpp-example-companyA-017.xml>

3591 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpp-example-companyB-017.xml>

3592  
3593 **draft-cpp-example-companyA-017.xml:**  
3594

```
3595 <?xml version="1.0"?>
3596 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
3597 <tp:CollaborationProtocolProfile
3598   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
3599   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3600   xmlns:xlink="http://www.w3.org/1999/xlink"
3601   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3602   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3603   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
3604     draft-cpp-cpa-017.xsd"
3605   tp:cppid="uri:companyA-cpp" tp:version="017">
3606 <!-- Party info for CompanyA-->
3607 <tp:PartyInfo
3608   tp:partyName="CompanyA"
3609   tp:defaultMshChannelId="asyncChannelA1"
3610   tp:defaultMshPackageId="CompanyA_MshSignalPackage">
3611 <tp:PartyId
3612   tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">123456789</tp:PartyId>
3613 <tp:PartyRef xlink:href="http://CompanyA.com/about.html"/>
3614 <tp:CollaborationRole>
3615 <tp:ProcessSpecification
3616   tp:version="2.0"
3617   tp:name="PIP3A4RequestPurchaseOrder"
3618   xlink:type="simple"
3619   xlink:href="http://www.rosettanet.org/processes/3A4.xml"
3620   tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
3621 <tp:Role
3622   tp:name="Buyer"
3623   xlink:type="simple"
3624   xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
3625 <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
3626 <tp:ServiceBinding>
3627 <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
3628 <tp:CanSend>
3629 <tp:ThisPartyActionBinding
3630   tp:id="companyA_ABID1"
3631   tp:action="Purchase Order Request Action"
3632   tp:packageId="CompanyA_RequestPackage">
3633 <tp:BusinessTransactionCharacteristics
3634   tp:isNonRepudiationRequired="true"
3635   tp:isNonRepudiationReceiptRequired="true"
3636   tp:isSecureTransportRequired="true"
3637   tp:isConfidential="transient"
3638   tp:isAuthenticated="persistent"
3639   tp:isTamperProof="persistent"
3640   tp:isAuthorizationRequired="true"
3641   tp:timeToAcknowledgeReceipt="PT2H" tp:timeToPerform="P1D"/>
3642 <tp>ActionContext
3643   tp:binaryCollaboration="Request Purchase Order"
3644   tp:businessTransactionActivity="Request Purchase Order"
3645   tp:requestOrResponseAction="Purchase Order Request Action"/>
3646 <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3647 </tp:ThisPartyActionBinding>
3648 </tp:CanSend>
3649 <tp:CanSend>
3650 <tp:ThisPartyActionBinding
3651   tp:id="companyA_ABID2"
```



```

3652         tp:action="ReceiptAcknowledgement "
3653         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
3654         <tp:BusinessTransactionCharacteristics
3655             tp:isNonRepudiationRequired="true"
3656             tp:isNonRepudiationReceiptRequired="true"
3657             tp:isSecureTransportRequired="true"
3658             tp:isConfidential="transient"
3659             tp:isAuthenticated="persistent"
3660             tp:isTamperProof="persistent"
3661             tp:isAuthorizationRequired="true"
3662             tp:timeToAcknowledgeReceipt="PT2H"
3663             tp:timeToPerform="P1D" />
3664         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3665     </tp:ThisPartyActionBinding>
3666 </tp:CanSend>
3667 <!-- The next binding uses a synchronous delivery channel -->
3668 <tp:CanSend>
3669     <tp:ThisPartyActionBinding
3670         tp:id="companyA_ABID6"
3671         tp:action="Purchase Order Request Action"
3672         tp:packageId="CompanyA_RequestPackage">
3673         <tp:BusinessTransactionCharacteristics
3674             tp:isNonRepudiationRequired="true"
3675             tp:isNonRepudiationReceiptRequired="true"
3676             tp:isSecureTransportRequired="true "
3677             tp:isConfidential="transient"
3678             tp:isAuthenticated="persistent"
3679             tp:isTamperProof="persistent"
3680             tp:isAuthorizationRequired="true"
3681             tp:timeToAcknowledgeReceipt="PT5M"
3682             tp:timeToPerform="PT5M" />
3683         <tp>ActionContext
3684             tp:binaryCollaboration="Request Purchase Order"
3685             tp:businessTransactionActivity="Request Purchase Order"
3686             tp:requestOrResponseAction="Purchase Order Request Action"/>
3687         <tp:ChannelId>syncChannelA1</tp:ChannelId>
3688     </tp:ThisPartyActionBinding>
3689 <tp:CanReceive>
3690     <tp:ThisPartyActionBinding
3691         tp:id="companyA_ABID7"
3692         tp:action="Purchase Order Confirmation Action"
3693         tp:packageId="CompanyA_SyncReplyPackage">
3694         <tp:BusinessTransactionCharacteristics
3695             tp:isNonRepudiationRequired="true"
3696             tp:isNonRepudiationReceiptRequired="true"
3697             tp:isSecureTransportRequired="true"
3698             tp:isConfidential="transient"
3699             tp:isAuthenticated="persistent"
3700             tp:isTamperProof="persistent"
3701             tp:isAuthorizationRequired="true"
3702             tp:timeToAcknowledgeReceipt="PT5M"
3703             tp:timeToPerform="PT5M" />
3704         <tp>ActionContext
3705             tp:binaryCollaboration="Request Purchase Order"
3706             tp:businessTransactionActivity="Request Purchase Order"
3707             tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
3708         <tp:ChannelId>syncChannelA1</tp:ChannelId>
3709     </tp:ThisPartyActionBinding>
3710 </tp:CanReceive>
3711 <tp:CanReceive>
3712     <tp:ThisPartyActionBinding
3713         tp:id="companyA_ABID8"
3714         tp:action="Exception"
3715         tp:packageId="CompanyA_ExceptionPackage">
3716         <tp:BusinessTransactionCharacteristics
3717             tp:isNonRepudiationRequired="true"
3718             tp:isNonRepudiationReceiptRequired="true"
3719             tp:isSecureTransportRequired="true"
3720             tp:isConfidential="transient"
3721             tp:isAuthenticated="persistent"
3722             tp:isTamperProof="persistent"

```

```

3723         tp:isAuthorizationRequired="true"
3724         tp:timeToAcknowledgeReceipt="PT5M"
3725         tp:timeToPerform="PT5M" />
3726     <tp:ChannelId>syncChannelA1</tp:ChannelId>
3727 </tp:ThisPartyActionBinding>
3728 </tp:CanReceive>
3729 </tp:CanSend>
3730 <tp:CanReceive>
3731 <tp:ThisPartyActionBinding
3732     tp:id="companyA_ABID3"
3733     tp:action="Purchase Order Confirmation Action"
3734     tp:packageId="CompanyA_ResponsePackage">
3735     <tp:BusinessTransactionCharacteristics
3736         tp:isNonRepudiationRequired="true"
3737         tp:isNonRepudiationReceiptRequired="true"
3738         tp:isSecureTransportRequired="true"
3739         tp:isConfidential="transient"
3740         tp:isAuthenticated="persistent"
3741         tp:isTamperProof="persistent"
3742         tp:isAuthorizationRequired="true"
3743         tp:timeToAcknowledgeReceipt="PT2H"
3744         tp:timeToPerform="P1D" />
3745     <tp:ActionContext
3746         tp:binaryCollaboration="Request Purchase Order"
3747         tp:businessTransactionActivity="Request Purchase Order"
3748         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
3749     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3750 </tp:ThisPartyActionBinding>
3751 </tp:CanReceive>
3752 <tp:CanReceive>
3753 <tp:ThisPartyActionBinding
3754     tp:id="companyA_ABID4"
3755     tp:action="ReceiptAcknowledgment"
3756     tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
3757     <tp:BusinessTransactionCharacteristics
3758         tp:isNonRepudiationRequired="true"
3759         tp:isNonRepudiationReceiptRequired="true"
3760         tp:isSecureTransportRequired="true"
3761         tp:isConfidential="transient"
3762         tp:isAuthenticated="persistent" tp:isTamperProof="persistent"
3763         tp:isAuthorizationRequired="true"
3764         tp:timeToAcknowledgeReceipt="PT2H"
3765         tp:timeToPerform="P1D" />
3766     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3767 </tp:ThisPartyActionBinding>
3768 </tp:CanReceive>
3769 <tp:CanReceive>
3770 <tp:ThisPartyActionBinding
3771     tp:id="companyA_ABID5"
3772     tp:action="Exception"
3773     tp:packageId="CompanyA_ExceptionPackage">
3774     <tp:BusinessTransactionCharacteristics
3775         tp:isNonRepudiationRequired="true"
3776         tp:isNonRepudiationReceiptRequired="true"
3777         tp:isSecureTransportRequired="true"
3778         tp:isConfidential="transient"
3779         tp:isAuthenticated="persistent"
3780         tp:isTamperProof="persistent"
3781         tp:isAuthorizationRequired="true"
3782         tp:timeToAcknowledgeReceipt="PT2H"
3783         tp:timeToPerform="P1D" />
3784     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3785 </tp:ThisPartyActionBinding>
3786 </tp:CanReceive>
3787 </tp:ServiceBinding>
3788 </tp:CollaborationRole>
3789 <!-- Certificates used by the "Buyer" company -->
3790 <tp:Certificate tp:certId="CompanyA_AppCert">
3791     <ds:KeyInfo
3792         <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
3793     </ds:KeyInfo>

```

```

3794     </tp:Certificate>
3795     <tp:Certificate tp:certId="CompanyA_SigningCert">
3796         <ds:KeyInfo>
3797             <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
3798         </ds:KeyInfo>
3799     </tp:Certificate>
3800     <tp:Certificate tp:certId="CompanyA_EncryptionCert">
3801         <ds:KeyInfo>
3802             <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
3803         </ds:KeyInfo>
3804     </tp:Certificate>
3805     <tp:Certificate tp:certId="CompanyA_ServerCert">
3806         <ds:KeyInfo>
3807             <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
3808         </ds:KeyInfo>
3809     </tp:Certificate>
3810     <tp:Certificate tp:certId="CompanyA_ClientCert">
3811         <ds:KeyInfo>
3812             <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
3813         </ds:KeyInfo>
3814     </tp:Certificate>
3815     <tp:Certificate tp:certId="TrustedRootCertA1">
3816         <ds:KeyInfo>
3817             <ds:KeyName>TrustedRootCertA1_Key</ds:KeyName>
3818         </ds:KeyInfo>
3819     </tp:Certificate>
3820     <tp:Certificate tp:certId="TrustedRootCertA2">
3821         <ds:KeyInfo>
3822             <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
3823         </ds:KeyInfo>
3824     </tp:Certificate>
3825     <tp:Certificate tp:certId="TrustedRootCertA3">
3826         <ds:KeyInfo>
3827             <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
3828         </ds:KeyInfo>
3829     </tp:Certificate>
3830     <tp:Certificate tp:certId="TrustedRootCertA4">
3831         <ds:KeyInfo>
3832             <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
3833         </ds:KeyInfo>
3834     </tp:Certificate>
3835     <tp:Certificate tp:certId="TrustedRootCertA5">
3836         <ds:KeyInfo>
3837             <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
3838         </ds:KeyInfo>
3839     </tp:Certificate>
3840     <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
3841         <tp:TrustAnchors>
3842             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1"/>
3843             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2"/>
3844             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4"/>
3845         </tp:TrustAnchors>
3846     </tp:SecurityDetails>
3847     <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
3848         <tp:TrustAnchors>
3849             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
3850             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
3851         </tp:TrustAnchors>
3852     </tp:SecurityDetails>
3853     <!-- An asynchronous delivery channel -->
3854     <tp:DeliveryChannel
3855         tp:channelId="asyncChannelA1"
3856         tp:transportId="transportA2"
3857         tp:docExchangeId="docExchangeA1">
3858         <tp:MessagingCharacteristics
3859             tp:syncReplyMode="none"
3860             tp:ackRequested="always"
3861             tp:ackSignatureRequested="always"
3862             tp:duplicateElimination="always"/>
3863     </tp:DeliveryChannel>
3864     <!-- A synchronous delivery channel -->

```

```

3865 <tp:DeliveryChannel
3866   tp:channelId="syncChannelA1"
3867   tp:transportId="transportA1"
3868   tp:docExchangeId="docExchangeA1">
3869   <tp:MessagingCharacteristics
3870     tp:syncReplyMode="none"
3871     tp:ackRequested="always"
3872     tp:ackSignatureRequested="always"
3873     tp:duplicateElimination="always"/>
3874 </tp:DeliveryChannel>
3875 <tp:Transport tp:transportId="transportA1">
3876   <tp:TransportSender>
3877     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3878     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3879     <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3880     <tp:TransportClientSecurity>
3881       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3882       <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
3883       <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3884     </tp:TransportClientSecurity>
3885   </tp:TransportSender>
3886   <tp:TransportReceiver>
3887     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3888     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3889     <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3890     <tp:Endpoint
3891       tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
3892       tp:type="allPurpose"/>
3893     <tp:TransportServerSecurity>
3894       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3895       <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
3896       <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3897     </tp:TransportServerSecurity>
3898   </tp:TransportReceiver>
3899 </tp:Transport>
3900 <tp:Transport tp:transportId="transportA2">
3901   <tp:TransportSender>
3902     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3903     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3904     <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3905     <tp:TransportClientSecurity>
3906       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3907       <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
3908       <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3909     </tp:TransportClientSecurity>
3910   </tp:TransportSender>
3911   <tp:TransportReceiver>
3912     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3913     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3914     <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3915     <tp:Endpoint
3916       tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
3917       tp:type="allPurpose"/>
3918     <tp:TransportServerSecurity>
3919       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3920       <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
3921       <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3922     </tp:TransportServerSecurity>
3923   </tp:TransportReceiver>
3924 </tp:Transport>
3925 <tp:DocExchange tp:docExchangeId="docExchangeA1">
3926   <tp:ebXMLSenderBinding tp:version="2.0">
3927     <tp:ReliableMessaging>
3928       <tp:Retries>3</tp:Retries>
3929       <tp:RetryInterval>PT2H</tp:RetryInterval>
3930       <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
3931     </tp:ReliableMessaging>
3932     <tp:PersistDuration>P1D</tp:PersistDuration>
3933     <tp:SenderNonRepudiation>
3934   </tp:SenderNonRepudiation>
3935 </tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>

```

```

3936         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
3937         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
3938     sha1</tp:SignatureAlgorithm>
3939         <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
3940     </tp:SenderNonRepudiation>
3941     <tp:SenderDigitalEnvelope>
3942         <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
3943         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
3944         <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
3945     </tp:SenderDigitalEnvelope>
3946 </tp:ebXMLSenderBinding>
3947 <tp:ebXMLReceiverBinding tp:version="2.0">
3948     <tp:ReliableMessaging>
3949         <tp:Retries>3</tp:Retries>
3950         <tp:RetryInterval>PT2H</tp:RetryInterval>
3951         <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
3952     </tp:ReliableMessaging>
3953     <tp:PersistDuration>P1D</tp:PersistDuration>
3954     <tp:ReceiverNonRepudiation>
3955
3956 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
3957     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
3958     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
3959     sha1</tp:SignatureAlgorithm>
3960         <tp:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
3961     </tp:ReceiverNonRepudiation>
3962     <tp:ReceiverDigitalEnvelope>
3963         <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
3964         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
3965         <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
3966     </tp:ReceiverDigitalEnvelope>
3967 </tp:ebXMLReceiverBinding>
3968 </tp:DocExchange>
3969 </tp:PartyInfo>
3970 <!-- SimplePart corresponding to the SOAP Envelope -->
3971 <tp:SimplePart
3972     tp:id="CompanyA_MsgHdr"
3973     tp:mimetype="text/xml">
3974     <tp:NamespaceSupported
3975         tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
3976         tp:version="2.0">
3977         http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
3978     </tp:NamespaceSupported>
3979 </tp:SimplePart>
3980 <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
3981 <tp:SimplePart
3982     tp:id="CompanyA_ReceiptAcknowledgment"
3983     tp:mimetype="application/xml">
3984     <tp:NamespaceSupported
3985         tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
3986         tp:version="2.0">
3987         http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
3988     </tp:NamespaceSupported>
3989 </tp:SimplePart>
3990 <!-- SimplePart corresponding to an Exception business signal -->
3991 <tp:SimplePart
3992     tp:id="CompanyA_Exception"
3993     tp:mimetype="application/xml">
3994     <tp:NamespaceSupported
3995         tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
3996         tp:version="2.0">
3997         http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
3998     </tp:NamespaceSupported>
3999 </tp:SimplePart>
4000 <!-- SimplePart corresponding to a request action -->
4001 <tp:SimplePart
4002     tp:id="CompanyA_Request"
4003     tp:mimetype="application/xml">
4004     <tp:NamespaceSupported
4005         tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
4006         tp:version="2.0">

```

```
4007     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
4008   </tp:NamespaceSupported>
4009 </tp:SimplePart>
4010 <!-- SimplePart corresponding to a response action -->
4011 <tp:SimplePart
4012   tp:id="CompanyA_Response"
4013   tp:mimetype="application/xml">
4014   <tp:NamespaceSupported
4015     tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
4016     tp:version="2.0">
4017     http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
4018   </tp:NamespaceSupported>
4019 </tp:SimplePart>
4020 <!-- An ebXML message with a SOAP Envelope only -->
4021 <tp:Packaging tp:id="CompanyA_MshSignalPackage">
4022   <tp:ProcessingCapabilities
4023     tp:parse="true"
4024     tp:generate="true"/>
4025   <tp:CompositeList>
4026     <tp:Composite
4027       tp:id="CompanyA_MshSignal"
4028       tp:mimetype="multipart/related"
4029       tp:mimeparameters="type=text/xml">
4030       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4031     </tp:Composite>
4032   </tp:CompositeList>
4033 </tp:Packaging>
4034 <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
4035 <tp:Packaging tp:id="CompanyA_RequestPackage">
4036   <tp:ProcessingCapabilities
4037     tp:parse="true"
4038     tp:generate="true"/>
4039   <tp:CompositeList>
4040     <tp:Composite
4041       tp:id="CompanyA_RequestMsg"
4042       tp:mimetype="multipart/related"
4043       tp:mimeparameters="type=text/xml">
4044       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4045       <tp:Constituent tp:idref="CompanyA_Request"/>
4046     </tp:Composite>
4047   </tp:CompositeList>
4048 </tp:Packaging>
4049 <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
4050 <tp:Packaging tp:id="CompanyA_ResponsePackage">
4051   <tp:ProcessingCapabilities
4052     tp:parse="true"
4053     tp:generate="true"/>
4054   <tp:CompositeList>
4055     <tp:Composite
4056       tp:id="CompanyA_ResponseMsg"
4057       tp:mimetype="multipart/related"
4058       tp:mimeparameters="type=text/xml">
4059       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4060       <tp:Constituent tp:idref="CompanyA_Response"/>
4061     </tp:Composite>
4062   </tp:CompositeList>
4063 </tp:Packaging>
4064 <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
4065   or an ebXML message with an Exception signal -->
4066 <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
4067   <tp:ProcessingCapabilities
4068     tp:parse="true"
4069     tp:generate="true"/>
4070   <tp:CompositeList>
4071     <tp:Composite
4072       tp:id="CompanyA_SignalAndResponseMsg"
4073       tp:mimetype="multipart/related"
4074       tp:mimeparameters="type=text/xml">
4075       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4076       <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
4077       <tp:Constituent tp:idref="CompanyA_Response"/>
```

```

4078     </tp:Composite>
4079   </tp:CompositeList>
4080 </tp:Packaging>
4081 <!-- An ebXML message with a SOAP Envelope plus a ReceiptAcknowledgment payload -->
4082 <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
4083   <tp:ProcessingCapabilities
4084     tp:parse="true"
4085     tp:generate="true"/>
4086   <tp:CompositeList>
4087     <tp:Composite
4088       tp:id="CompanyA_ReceiptAcknowledgmentMsg"
4089       tp:mimetype="multipart/related"
4090       tp:mimeparameters="type=text/xml">
4091       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4092       <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
4093     </tp:Composite>
4094   </tp:CompositeList>
4095 </tp:Packaging>
4096 <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
4097 <tp:Packaging tp:id="CompanyA_ExceptionPackage">
4098   <tp:ProcessingCapabilities
4099     tp:parse="true"
4100     tp:generate="true"/>
4101   <tp:CompositeList>
4102     <tp:Composite
4103       tp:id="CompanyA_ExceptionMsg"
4104       tp:mimetype="multipart/related"
4105       tp:mimeparameters="type=text/xml">
4106       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4107       <tp:Constituent tp:idref="CompanyA_Exception"/>
4108     </tp:Composite>
4109   </tp:CompositeList>
4110 </tp:Packaging>
4111 <tp:Comment xml:lang="en-US">Buyer's Collaboration Protocol Profile</tp:Comment>
4112 </tp:CollaborationProtocolProfile>
4113
4114
4115 draft-cpp-example-companyB-017.xml:
4116
4117 <?xml version="1.0"?>
4118 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
4119 <tp:CollaborationProtocolProfile
4120   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
4121   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4122   xmlns:xlink="http://www.w3.org/1999/xlink"
4123   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4124   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4125   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
4126     draft-cpp-cpa-017.xsd"
4127   tp:cppid="uri:companyB-cpp"
4128   tp:version="017">
4129 <!-- Party info for CompanyB-->
4130 <tp:PartyInfo
4131   tp:partyName="CompanyB"
4132   tp:defaultMshChannelId="asyncChannelB1"
4133   tp:defaultMshPackageId="CompanyB_MshSignalPackage">
4134 <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
4135 <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html"/>
4136 <tp:CollaborationRole>
4137   <tp:ProcessSpecification
4138     tp:version="2.0"
4139     tp:name="PIP3A4RequestPurchaseOrder"
4140     xlink:type="simple" xlink:href="http://www.rosettanet.org/processes/3A4.xml"
4141     tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
4142   <tp:Role
4143     tp:name="Seller"
4144     xlink:type="simple"
4145     xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
4146   <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert"/>
4147 <tp:ServiceBinding>
4148   <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>

```

```

4149     <tp:CanSend>
4150         <tp:ThisPartyActionBinding
4151             tp:id="companyB_ABID1"
4152             tp:action="Purchase Order Confirmation Action"
4153             tp:packageId="CompanyB_ResponsePackage">
4154             <tp:BusinessTransactionCharacteristics
4155                 tp:isNonRepudiationRequired="true"
4156                 tp:isNonRepudiationReceiptRequired="true"
4157                 tp:isSecureTransportRequired="true"
4158                 tp:isConfidential="transient"
4159                 tp:isAuthenticated="persistent"
4160                 tp:isTamperProof="persistent"
4161                 tp:isAuthorizationRequired="true"
4162                 tp:timeToAcknowledgeReceipt="PT2H"
4163                 tp:timeToPerform="P1D" />
4164             <tp>ActionContext
4165                 tp:binaryCollaboration="Request Purchase Order"
4166                 tp:businessTransactionActivity="Request Purchase Order"
4167                 tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4168             <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4169         </tp:ThisPartyActionBinding>
4170     </tp:CanSend>
4171     <tp:CanSend>
4172         <tp:ThisPartyActionBinding
4173             tp:id="companyB_ABID2"
4174             tp:action="ReceiptAcknowledgement"
4175             tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
4176             <tp:BusinessTransactionCharacteristics
4177                 tp:isNonRepudiationRequired="true"
4178                 tp:isNonRepudiationReceiptRequired="true"
4179                 tp:isSecureTransportRequired="true"
4180                 tp:isConfidential="transient"
4181                 tp:isAuthenticated="persistent"
4182                 tp:isTamperProof="persistent"
4183                 tp:isAuthorizationRequired="true"
4184                 tp:timeToAcknowledgeReceipt="PT2H"
4185                 tp:timeToPerform="P1D" />
4186             <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4187         </tp:ThisPartyActionBinding>
4188     </tp:CanSend>
4189     <tp:CanSend>
4190         <tp:ThisPartyActionBinding
4191             tp:id="companyB_ABID3"
4192             tp:action="Exception"
4193             tp:packageId="CompanyB_ExceptionPackage">
4194             <tp:BusinessTransactionCharacteristics
4195                 tp:isNonRepudiationRequired="true"
4196                 tp:isNonRepudiationReceiptRequired="true"
4197                 tp:isSecureTransportRequired="true"
4198                 tp:isConfidential="transient"
4199                 tp:isAuthenticated="persistent"
4200                 tp:isTamperProof="persistent"
4201                 tp:isAuthorizationRequired="true"
4202                 tp:timeToAcknowledgeReceipt="PT2H"
4203                 tp:timeToPerform="P1D" />
4204             <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4205         </tp:ThisPartyActionBinding>
4206     </tp:CanSend>
4207     <tp:CanReceive>
4208         <tp:ThisPartyActionBinding
4209             tp:id="companyB_ABID4"
4210             tp:action="Purchase Order Request Action"
4211             tp:packageId="CompanyB_RequestPackage">
4212             <tp:BusinessTransactionCharacteristics
4213                 tp:isNonRepudiationRequired="true"
4214                 tp:isNonRepudiationReceiptRequired="true"
4215                 tp:isSecureTransportRequired="true"
4216                 tp:isConfidential="transient"
4217                 tp:isAuthenticated="persistent"
4218                 tp:isTamperProof="persistent"
4219                 tp:isAuthorizationRequired="true"

```



```

4220         tp:timeToAcknowledgeReceipt="PT2H"
4221         tp:timeToPerform="P1D" />
4222     <tp:ActionContext
4223         tp:binaryCollaboration="Request Purchase Order"
4224         tp:businessTransactionActivity="Request Purchase Order"
4225         tp:requestOrResponseAction="Purchase Order Request Action"/>
4226     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4227 </tp:ThisPartyActionBinding>
4228 </tp:CanReceive>
4229 <tp:CanReceive>
4230     <tp:ThisPartyActionBinding
4231         tp:id="companyB_ABID5" tp:action="ReceiptAcknowledgment"
4232         tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
4233     <tp:BusinessTransactionCharacteristics
4234         tp:isNonRepudiationRequired="true"
4235         tp:isNonRepudiationReceiptRequired="true"
4236         tp:isSecureTransportRequired="true"
4237         tp:isConfidential="transient"
4238         tp:isAuthenticated="persistent"
4239         tp:isTamperProof="persistent"
4240         tp:isAuthorizationRequired="true"
4241         tp:timeToAcknowledgeReceipt="PT2H"
4242         tp:timeToPerform="P1D" />
4243     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4244 </tp:ThisPartyActionBinding>
4245 </tp:CanReceive>
4246 <!-- The next binding uses a synchronous delivery channel -->
4247 <tp:CanReceive>
4248     <tp:ThisPartyActionBinding
4249         tp:id="companyB_ABID8"
4250         tp:action="Purchase Order Request Action"
4251         tp:packageId="CompanyB_SyncReplyPackage">
4252     <tp:BusinessTransactionCharacteristics
4253         tp:isNonRepudiationRequired="true"
4254         tp:isNonRepudiationReceiptRequired="true"
4255         tp:isSecureTransportRequired="true"
4256         tp:isConfidential="transient"
4257         tp:isAuthenticated="persistent"
4258         tp:isTamperProof="persistent"
4259         tp:isAuthorizationRequired="true"
4260         tp:timeToAcknowledgeReceipt="PT5M"
4261         tp:timeToPerform="PT5M" />
4262     <tp:ActionContext
4263         tp:binaryCollaboration="Request Purchase Order"
4264         tp:businessTransactionActivity="Request Purchase Order"
4265         tp:requestOrResponseAction="Purchase Order Request Action"/>
4266     <tp:ChannelId>syncChannelB1</tp:ChannelId>
4267 </tp:ThisPartyActionBinding>
4268 <tp:CanSend>
4269     <tp:ThisPartyActionBinding
4270         tp:id="companyB_ABID6"
4271         tp:action="Purchase Order Confirmation Action"
4272         tp:packageId="CompanyB_ResponsePackage">
4273     <tp:BusinessTransactionCharacteristics
4274         tp:isNonRepudiationRequired="true"
4275         tp:isNonRepudiationReceiptRequired="true"
4276         tp:isSecureTransportRequired="true"
4277         tp:isConfidential="transient"
4278         tp:isAuthenticated="persistent"
4279         tp:isTamperProof="persistent"
4280         tp:isAuthorizationRequired="true"
4281         tp:timeToAcknowledgeReceipt="PT5M"
4282         tp:timeToPerform="PT5M" />
4283     <tp:ActionContext
4284         tp:binaryCollaboration="Request Purchase Order"
4285         tp:businessTransactionActivity="Request Purchase Order"
4286         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4287     <tp:ChannelId>syncChannelB1</tp:ChannelId>
4288 </tp:ThisPartyActionBinding>
4289 </tp:CanSend>
4290 <tp:CanSend>

```

```

4291         <tp:ThisPartyActionBinding
4292             tp:id="companyB_ABID7"
4293             tp:action="Exception"
4294             tp:packageId="CompanyB_ExceptionPackage">
4295             <tp:BusinessTransactionCharacteristics
4296                 tp:isNonRepudiationRequired="true"
4297                 tp:isNonRepudiationReceiptRequired="true"
4298                 tp:isSecureTransportRequired="true"
4299                 tp:isConfidential="transient"
4300                 tp:isAuthenticated="persistent"
4301                 tp:isTamperProof="persistent"
4302                 tp:isAuthorizationRequired="true"
4303                 tp:timeToAcknowledgeReceipt="PT5M"
4304                 tp:timeToPerform="PT5M" />
4305             <tp:ChannelId>syncChannelB1</tp:ChannelId>
4306         </tp:ThisPartyActionBinding>
4307     </tp:CanSend>
4308 </tp:CanReceive>
4309 </tp:ServiceBinding>
4310 </tp:CollaborationRole>
4311 <!-- Certificates used by the "Seller" company -->
4312 <tp:Certificate tp:certId="CompanyB_AppCert">
4313     <ds:KeyInfo>
4314         <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
4315     </ds:KeyInfo>
4316 </tp:Certificate>
4317 <tp:Certificate tp:certId="CompanyB_SigningCert">
4318     <ds:KeyInfo>
4319         <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
4320     </ds:KeyInfo>
4321 </tp:Certificate>
4322 <tp:Certificate tp:certId="CompanyB_EncryptionCert">
4323     <ds:KeyInfo>
4324         <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
4325     </ds:KeyInfo>
4326 </tp:Certificate>
4327 <tp:Certificate tp:certId="CompanyB_ServerCert">
4328     <ds:KeyInfo>
4329         <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
4330     </ds:KeyInfo>
4331 </tp:Certificate>
4332 <tp:Certificate tp:certId="CompanyB_ClientCert">
4333     <ds:KeyInfo>
4334         <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
4335     </ds:KeyInfo>
4336 </tp:Certificate>
4337 <tp:Certificate tp:certId="TrustedRootCertB4">
4338     <ds:KeyInfo>
4339         <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
4340     </ds:KeyInfo>
4341 </tp:Certificate>
4342 <tp:Certificate tp:certId="TrustedRootCertB5">
4343     <ds:KeyInfo>
4344         <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
4345     </ds:KeyInfo>
4346 </tp:Certificate>
4347 <tp:Certificate tp:certId="TrustedRootCertB6">
4348     <ds:KeyInfo>
4349         <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
4350     </ds:KeyInfo>
4351 </tp:Certificate>
4352 <tp:Certificate tp:certId="TrustedRootCertB7">
4353     <ds:KeyInfo>
4354         <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
4355     </ds:KeyInfo>
4356 </tp:Certificate>
4357 <tp:Certificate tp:certId="TrustedRootCertB8">
4358     <ds:KeyInfo>
4359         <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
4360     </ds:KeyInfo>
4361 </tp:Certificate>

```

```

4362     <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
4363         <tp:TrustAnchors>
4364             <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5" />
4365             <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6" />
4366             <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4" />
4367         </tp:TrustAnchors>
4368     </tp:SecurityDetails>
4369     <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
4370         <tp:TrustAnchors>
4371             <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8" />
4372             <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7" />
4373         </tp:TrustAnchors>
4374     </tp:SecurityDetails>
4375     <!-- An asynchronous delivery channel -->
4376     <tp:DeliveryChannel
4377         tp:channelId="asyncChannelB1"
4378         tp:transportId="transportB1"
4379         tp:docExchangeId="docExchangeB1">
4380         <tp:MessagingCharacteristics
4381             tp:syncReplyMode="none"
4382             tp:ackRequested="always"
4383             tp:ackSignatureRequested="always"
4384             tp:duplicateElimination="always" />
4385     </tp:DeliveryChannel>
4386     <!-- A synchronous delivery channel -->
4387     <tp:DeliveryChannel
4388         tp:channelId="syncChannelB1"
4389         tp:transportId="transportB2"
4390         tp:docExchangeId="docExchangeB1">
4391         <tp:MessagingCharacteristics
4392             tp:syncReplyMode="signalsAndResponse"
4393             tp:ackRequested="always"
4394             tp:ackSignatureRequested="always"
4395             tp:duplicateElimination="always" />
4396     </tp:DeliveryChannel>
4397     <tp:Transport tp:transportId="transportB1">
4398         <tp:TransportSender>
4399             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4400             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4401             <tp:TransportClientSecurity>
4402                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4403                 <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert" />
4404                 <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity" />
4405             </tp:TransportClientSecurity>
4406         </tp:TransportSender>
4407         <tp:TransportReceiver>
4408             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4409             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4410             <tp:Endpoint
4411                 tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
4412                 tp:type="allPurpose" />
4413             <tp:TransportServerSecurity>
4414                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4415                 <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert" />
4416                 <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity" />
4417             </tp:TransportServerSecurity>
4418         </tp:TransportReceiver>
4419     </tp:Transport>
4420     <tp:Transport tp:transportId="transportB2">
4421         <tp:TransportSender>
4422             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4423             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4424             <tp:TransportClientSecurity>
4425                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4426                 <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert" />
4427                 <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity" />
4428             </tp:TransportClientSecurity>
4429         </tp:TransportSender>
4430         <tp:TransportReceiver>
4431             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4432             <tp:AccessAuthentication>basic</tp:AccessAuthentication>

```

```

4433     <tp:Endpoint
4434         tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
4435         tp:type="allPurpose"/>
4436     <tp:TransportServerSecurity>
4437         <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4438         <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
4439         <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4440     </tp:TransportServerSecurity>
4441 </tp:TransportReceiver>
4442 </tp:Transport>
4443 <tp:DocExchange tp:docExchangeId="docExchangeB1">
4444     <tp:ebXMLSenderBinding tp:version="2.0">
4445         <tp:ReliableMessaging>
4446             <tp:Retries>3</tp:Retries>
4447             <tp:RetryInterval>PT2H</tp:RetryInterval>
4448             <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4449         </tp:ReliableMessaging>
4450         <tp:PersistDuration>P1D</tp:PersistDuration>
4451         <tp:SenderNonRepudiation>
4452 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4453         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4454         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4455         sha1</tp:SignatureAlgorithm>
4456         <tp:SigningCertificateRef tp:certId="CompanyB_SigningCert"/>
4457     </tp:SenderNonRepudiation>
4458     <tp:SenderDigitalEnvelope>
4459         <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4460         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4461         <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
4462     </tp:SenderDigitalEnvelope>
4463 </tp:ebXMLSenderBinding>
4464 <tp:ebXMLReceiverBinding tp:version="2.0">
4465     <tp:ReliableMessaging>
4466         <tp:Retries>3</tp:Retries>
4467         <tp:RetryInterval>PT2H</tp:RetryInterval>
4468         <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4469     </tp:ReliableMessaging>
4470     <tp:PersistDuration>P1D</tp:PersistDuration>
4471     <tp:ReceiverNonRepudiation>
4472 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4473         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4474         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4475         sha1</tp:SignatureAlgorithm>
4476         <tp:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
4477     </tp:ReceiverNonRepudiation>
4478     <tp:ReceiverDigitalEnvelope>
4479         <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4480         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4481         <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert"/>
4482     </tp:ReceiverDigitalEnvelope>
4483 </tp:ebXMLReceiverBinding>
4484 </tp:DocExchange>
4485 </tp:PartyInfo>
4486 <!-- SimplePart corresponding to the SOAP Envelope -->
4487 <tp:SimplePart
4488     tp:id="CompanyB_MsgHdr"
4489     tp:mimetype="text/xml">
4490     <tp:NamespaceSupported
4491         tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
4492         tp:version="2.0">
4493         http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
4494     </tp:NamespaceSupported>
4495 </tp:SimplePart>
4496 <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
4497 <tp:SimplePart
4498     tp:id="CompanyB_ReceiptAcknowledgment"
4499     tp:mimetype="application/xml">
4500     <tp:NamespaceSupported
4501         tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
4502     </tp:NamespaceSupported>
4503 </tp:SimplePart>

```

```

4504         tp:version="2.0">
4505         http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
4506     </tp:NamespaceSupported>
4507 </tp:SimplePart>
4508 <!-- SimplePart corresponding to an Exception business signal -->
4509 <tp:SimplePart
4510     tp:id="CompanyB_Exception"
4511     tp:mimetype="application/xml">
4512     <tp:NamespaceSupported
4513         tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
4514         tp:version="2.0">
4515         http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
4516     </tp:NamespaceSupported>
4517 </tp:SimplePart>
4518 <!-- SimplePart corresponding to a request action -->
4519 <tp:SimplePart
4520     tp:id="CompanyB_Request"
4521     tp:mimetype="application/xml">
4522     <tp:NamespaceSupported
4523         tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
4524         tp:version="2.0">
4525         http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
4526     </tp:NamespaceSupported>
4527 </tp:SimplePart>
4528 <!-- SimplePart corresponding to a response action -->
4529 <tp:SimplePart
4530     tp:id="CompanyB_Response"
4531     tp:mimetype="application/xml">
4532     <tp:NamespaceSupported
4533         tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd.xsd"
4534         tp:version="2.0">
4535         http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
4536     </tp:NamespaceSupported>
4537 </tp:SimplePart>
4538 <!-- An ebXML message with a SOAP Envelope only -->
4539 <tp:Packaging tp:id="CompanyB_MshSignalPackage">
4540     <tp:ProcessingCapabilities
4541         tp:parse="true"
4542         tp:generate="true"/>
4543     <tp:CompositeList>
4544         <tp:Composite
4545             tp:id="CompanyB_MshSignal"
4546             tp:mimetype="multipart/related"
4547             tp:mimeparameters="type=text/xml">
4548                 <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4549             </tp:Composite>
4550         </tp:CompositeList>
4551 </tp:Packaging>
4552 <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
4553 <tp:Packaging tp:id="CompanyB_RequestPackage">
4554     <tp:ProcessingCapabilities
4555         tp:parse="true"
4556         tp:generate="true"/>
4557     <tp:CompositeList>
4558         <tp:Composite
4559             tp:id="RequestMsg"
4560             tp:mimetype="multipart/related"
4561             tp:mimeparameters="type=text/xml">
4562                 <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4563                 <tp:Constituent tp:idref="CompanyB_Request"/>
4564             </tp:Composite>
4565         </tp:CompositeList>
4566 </tp:Packaging>
4567 <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
4568 <tp:Packaging tp:id="CompanyB_ResponsePackage">
4569     <tp:ProcessingCapabilities
4570         tp:parse="true"
4571         tp:generate="true"/>
4572     <tp:CompositeList>
4573         <tp:Composite
4574             tp:id="CompanyB_ResponseMsg"

```

```

4575         tp:mimetype="multipart/related"
4576         tp:mimeparameters="type=text/xml">
4577         <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4578         <tp:Constituent tp:idref="CompanyB_Response"/>
4579     </tp:Composite>
4580 </tp:CompositeList>
4581 </tp:Packaging>
4582 <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
4583 <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
4584     <tp:ProcessingCapabilities
4585         tp:parse="true"
4586         tp:generate="true"/>
4587     <tp:CompositeList>
4588         <tp:Composite
4589             tp:id="CompanyB_ReceiptAcknowledgmentMsg"
4590             tp:mimetype="multipart/related"
4591             tp:mimeparameters="type=text/xml">
4592             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4593             <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
4594         </tp:Composite>
4595     </tp:CompositeList>
4596 </tp:Packaging>
4597 <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
4598 <tp:Packaging tp:id="CompanyB_ExceptionPackage">
4599     <tp:ProcessingCapabilities
4600         tp:parse="true"
4601         tp:generate="true"/>
4602     <tp:CompositeList>
4603         <tp:Composite
4604             tp:id="CompanyB_ExceptionMsg"
4605             tp:mimetype="multipart/related"
4606             tp:mimeparameters="type=text/xml">
4607             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4608             <tp:Constituent tp:idref="CompanyB_Exception"/>
4609         </tp:Composite>
4610     </tp:CompositeList>
4611 </tp:Packaging>
4612 <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
4613     or an ebXML message with an Exception signal -->
4614 <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
4615     <tp:ProcessingCapabilities
4616         tp:parse="true"
4617         tp:generate="true"/>
4618     <tp:CompositeList>
4619         <tp:Composite
4620             tp:id="CompanyB_SignalAndResponseMsg"
4621             tp:mimetype="multipart/related"
4622             tp:mimeparameters="type=text/xml">
4623             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4624             <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
4625             <tp:Constituent tp:idref="CompanyB_Response"/>
4626         </tp:Composite>
4627     </tp:CompositeList>
4628     <tp:CompositeList>
4629         <tp:Composite
4630             tp:id="CompanyB_SyncExceptionMsg"
4631             tp:mimetype="multipart/related"
4632             tp:mimeparameters="type=text/xml">
4633             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4634             <tp:Constituent tp:idref="CompanyB_Exception"/>
4635         </tp:Composite>
4636     </tp:CompositeList>
4637 </tp:Packaging>
4638 <tp:Comment xml:lang="en-US">Seller's Collaboration Protocol Profile</tp:Comment>
4639 </tp:CollaborationProtocolProfile>
4640

```

## 4641 Appendix B Example of CPA Document (Non-Normative)

4642 The example in this appendix is to be parsed with an XML Schema parser. The schema is  
4643 available as an ASCII file at

4644 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpp-cpa-017.xsd>  
4645

4646 The example that can be parsed with the XSD is available at

4647 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpa-example-017.xml>  
4648

```
4649 <?xml version="1.0"?>
4650 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
4651 <tp:CollaborationProtocolAgreement
4652   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
4653   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4654   xmlns:xlink="http://www.w3.org/1999/xlink"
4655   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4656   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4657   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
4658                       draft-cpp-cpa-017.xsd"
4659   tp:cpaid="uri:companyA-and-companyB-cpa" tp:version="017">
4660   <tp:Status tp:value="proposed"/>
4661   <tp:Start>2001-05-20T07:21:00Z</tp:Start>
4662   <tp:End>2002-05-20T07:21:00Z</tp:End>
4663   <tp:ConversationConstraints tp:invocationLimit="100" tp:concurrentConversations="10"/>
4664   <!-- Party info for CompanyA -->
4665   <tp:PartyInfo
4666     tp:partyName="CompanyA"
4667     tp:defaultMshChannelId="asyncChannelA1"
4668     tp:defaultMshPackageId="CompanyA_MshSignalPackage">
4669     <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">123456789</tp:PartyId>
4670     <tp:PartyRef xlink:href="http://CompanyA.com/about.html"/>
4671     <tp:CollaborationRole>
4672       <tp:ProcessSpecification
4673         tp:version="2.0"
4674         tp:name="PIP3A4RequestPurchaseOrder "
4675         xlink:type="simple"
4676         xlink:href="http://www.rosettanet.org/processes/3A4.xml "
4677         tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
4678       <tp:Role
4679         tp:name="Buyer"
4680         xlink:type="simple"
4681         xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
4682       <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
4683       <tp:ServiceBinding>
4684         <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
4685         <tp:CanSend>
4686           <tp:ThisPartyActionBinding
4687             tp:id="companyA_ABID1"
4688             tp:action="Purchase Order Request Action"
4689             tp:packageId="CompanyA_RequestPackage">
4690             <tp:BusinessTransactionCharacteristics
4691               tp:isNonRepudiationRequired="true"
4692               tp:isNonRepudiationReceiptRequired="true"
4693               tp:isSecureTransportRequired="true"
4694               tp:isConfidential="transient"
4695               tp:isAuthenticated="persistent"
4696               tp:isTamperProof="persistent"
4697               tp:isAuthorizationRequired="true"
4698               tp:timeToAcknowledgeReceipt="PT2H"
4699               tp:timeToPerform="P1D"/>
4700             <tp:ActionContext
4701               tp:binaryCollaboration="Request Purchase Order"
4702               tp:businessTransactionActivity="Request Purchase Order"
4703               tp:requestOrResponseAction="Purchase Order Request Action"/>
4704             <tp:ChannelId>asyncChannelA1</tp:ChannelId>
```

```

4705         </tp:ThisPartyActionBinding>
4706         <tp:OtherPartyActionBinding>companyB_ABID4</tp:OtherPartyActionBinding>
4707     </tp:CanSend>
4708     <tp:CanSend>
4709         <tp:ThisPartyActionBinding
4710             tp:id="companyA_ABID2"
4711             tp:action="ReceiptAcknowledgement"
4712             tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
4713             <tp:BusinessTransactionCharacteristics
4714                 tp:isNonRepudiationRequired="true"
4715                 tp:isNonRepudiationReceiptRequired="true"
4716                 tp:isSecureTransportRequired="true"
4717                 tp:isConfidential="transient"
4718                 tp:isAuthenticated="persistent"
4719                 tp:isTamperProof="persistent"
4720                 tp:isAuthorizationRequired="true"
4721                 tp:timeToAcknowledgeReceipt="PT2H"
4722                 tp:timeToPerform="P1D"/>
4723             <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4724         </tp:ThisPartyActionBinding>
4725         <tp:OtherPartyActionBinding>companyB_ABID5</tp:OtherPartyActionBinding>
4726     </tp:CanSend>
4727     <!-- The next binding uses a synchronous delivery channel -->
4728     <tp:CanSend>
4729         <tp:ThisPartyActionBinding
4730             tp:id="companyA_ABID6"
4731             tp:action="Purchase Order Request Action"
4732             tp:packageId="CompanyA_RequestPackage">
4733             <tp:BusinessTransactionCharacteristics
4734                 tp:isNonRepudiationRequired="true"
4735                 tp:isNonRepudiationReceiptRequired="true"
4736                 tp:isSecureTransportRequired="true"
4737                 tp:isConfidential="transient"
4738                 tp:isAuthenticated="persistent"
4739                 tp:isTamperProof="persistent"
4740                 tp:isAuthorizationRequired="true"
4741                 tp:timeToAcknowledgeReceipt="PT5M"
4742                 tp:timeToPerform="PT5M"/>
4743             <tp>ActionContext
4744                 tp:binaryCollaboration="Request Purchase Order"
4745                 tp:businessTransactionActivity="Request Purchase Order"
4746                 tp:requestOrResponseAction="Purchase Order Request Action"/>
4747             <tp:ChannelId>syncChannelA1</tp:ChannelId>
4748         </tp:ThisPartyActionBinding>
4749         <tp:OtherPartyActionBinding>companyB_ABID6</tp:OtherPartyActionBinding>
4750     <tp:CanReceive>
4751         <tp:ThisPartyActionBinding
4752             tp:id="companyA_ABID7"
4753             tp:action="Purchase Order Confirmation Action"
4754             tp:packageId="CompanyA_SyncReplyPackage">
4755             <tp:BusinessTransactionCharacteristics
4756                 tp:isNonRepudiationRequired="true"
4757                 tp:isNonRepudiationReceiptRequired="true"
4758                 tp:isSecureTransportRequired="true"
4759                 tp:isConfidential="transient"
4760                 tp:isAuthenticated="persistent"
4761                 tp:isTamperProof="persistent"
4762                 tp:isAuthorizationRequired="true"
4763                 tp:timeToAcknowledgeReceipt="PT5M"
4764                 tp:timeToPerform="PT5M"/>
4765             <tp>ActionContext
4766                 tp:binaryCollaboration="Request Purchase Order"
4767                 tp:businessTransactionActivity="Request Purchase Order"
4768                 tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4769             <tp:ChannelId>syncChannelA1</tp:ChannelId>
4770         </tp:ThisPartyActionBinding>
4771         <tp:OtherPartyActionBinding>companyB_ABID7</tp:OtherPartyActionBinding>
4772     </tp:CanReceive>
4773     <tp:CanReceive>
4774         <tp:ThisPartyActionBinding
4775             tp:id="companyA_ABID8"

```



```

4776         tp:action="Exception"
4777         tp:packageId="CompanyA_ExceptionPackage">
4778         <tp:BusinessTransactionCharacteristics
4779             tp:isNonRepudiationRequired="true"
4780             tp:isNonRepudiationReceiptRequired="true"
4781             tp:isSecureTransportRequired="true"
4782             tp:isConfidential="transient"
4783             tp:isAuthenticated="persistent"
4784             tp:isTamperProof="persistent"
4785             tp:isAuthorizationRequired="true"
4786             tp:timeToAcknowledgeReceipt="PT5M"
4787             tp:timeToPerform="PT5M" />
4788         <tp:ChannelId>syncChannelA1</tp:ChannelId>
4789     </tp:ThisPartyActionBinding>
4790     <tp:OtherPartyActionBinding>companyB_ABID8</tp:OtherPartyActionBinding>
4791 </tp:CanReceive>
4792 </tp:CanSend>
4793 <tp:CanReceive>
4794     <tp:ThisPartyActionBinding
4795         tp:id="companyA_ABID3"
4796         tp:action="Purchase Order Confirmation Action"
4797         tp:packageId="CompanyA_ResponsePackage">
4798         <tp:BusinessTransactionCharacteristics
4799             tp:isNonRepudiationRequired="true"
4800             tp:isNonRepudiationReceiptRequired="true"
4801             tp:isSecureTransportRequired="true"
4802             tp:isConfidential="transient"
4803             tp:isAuthenticated="persistent"
4804             tp:isTamperProof="persistent"
4805             tp:isAuthorizationRequired="true"
4806             tp:timeToAcknowledgeReceipt="PT2H"
4807             tp:timeToPerform="P1D" />
4808         <tp:ActionContext
4809             tp:binaryCollaboration="Request Purchase Order"
4810             tp:businessTransactionActivity="Request Purchase Order"
4811             tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4812         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4813     </tp:ThisPartyActionBinding>
4814     <tp:OtherPartyActionBinding>companyB_ABID1</tp:OtherPartyActionBinding>
4815 </tp:CanReceive>
4816 <tp:CanReceive>
4817     <tp:ThisPartyActionBinding
4818         tp:id="companyA_ABID4"
4819         tp:action="ReceiptAcknowledgment"
4820         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
4821         <tp:BusinessTransactionCharacteristics
4822             tp:isNonRepudiationRequired="true"
4823             tp:isNonRepudiationReceiptRequired="true"
4824             tp:isSecureTransportRequired="true"
4825             tp:isConfidential="transient"
4826             tp:isAuthenticated="persistent"
4827             tp:isTamperProof="persistent"
4828             tp:isAuthorizationRequired="true"
4829             tp:timeToAcknowledgeReceipt="PT2H" tp:timeToPerform="P1D" />
4830         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4831     </tp:ThisPartyActionBinding>
4832     <tp:OtherPartyActionBinding>companyB_ABID2</tp:OtherPartyActionBinding>
4833 </tp:CanReceive>
4834 <tp:CanReceive>
4835     <tp:ThisPartyActionBinding
4836         tp:id="companyA_ABID5"
4837         tp:action="Exception"
4838         tp:packageId="CompanyA_ExceptionPackage">
4839         <tp:BusinessTransactionCharacteristics
4840             tp:isNonRepudiationRequired="true"
4841             tp:isNonRepudiationReceiptRequired="true"
4842             tp:isSecureTransportRequired="true"
4843             tp:isConfidential="transient"
4844             tp:isAuthenticated="persistent"
4845             tp:isTamperProof="persistent"
4846             tp:isAuthorizationRequired="true"

```

```

4847         tp:timeToAcknowledgeReceipt="PT2H"
4848         tp:timeToPerform="P1D" />
4849     </tp:ChannelId>asyncChannelA1</tp:ChannelId>
4850     </tp:ThisPartyActionBinding>
4851     <tp:OtherPartyActionBinding>companyB_ABID3</tp:OtherPartyActionBinding>
4852 </tp:CanReceive>
4853 </tp:ServiceBinding>
4854 </tp:CollaborationRole>
4855 <!-- Certificates used by the "Buyer" company -->
4856 <tp:Certificate tp:certId="CompanyA_AppCert">
4857     <ds:KeyInfo>
4858         <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
4859     </ds:KeyInfo>
4860 </tp:Certificate>
4861 <tp:Certificate tp:certId="CompanyA_SigningCert">
4862     <ds:KeyInfo>
4863         <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
4864     </ds:KeyInfo>
4865 </tp:Certificate>
4866 <tp:Certificate tp:certId="CompanyA_EncryptionCert">
4867     <ds:KeyInfo>
4868         <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
4869     </ds:KeyInfo>
4870 </tp:Certificate>
4871 <tp:Certificate tp:certId="CompanyA_ServerCert">
4872     <ds:KeyInfo>
4873         <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
4874     </ds:KeyInfo>
4875 </tp:Certificate>
4876 <tp:Certificate tp:certId="CompanyA_ClientCert">
4877     <ds:KeyInfo>
4878         <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
4879     </ds:KeyInfo>
4880 </tp:Certificate>
4881 <tp:Certificate tp:certId="TrustedRootCertA1">
4882     <ds:KeyInfo>
4883         <ds:KeyName>TrustedRootCertA1_Key </ds:KeyName>
4884     </ds:KeyInfo>
4885 </tp:Certificate>
4886 <tp:Certificate tp:certId="TrustedRootCertA2">
4887     <ds:KeyInfo>
4888         <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
4889     </ds:KeyInfo>
4890 </tp:Certificate>
4891 <tp:Certificate tp:certId="TrustedRootCertA3">
4892     <ds:KeyInfo>
4893         <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
4894     </ds:KeyInfo>
4895 </tp:Certificate>
4896 <tp:Certificate tp:certId="TrustedRootCertA4">
4897     <ds:KeyInfo>
4898         <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
4899     </ds:KeyInfo>
4900 </tp:Certificate>
4901 <tp:Certificate tp:certId="TrustedRootCertA5">
4902     <ds:KeyInfo>
4903         <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
4904     </ds:KeyInfo>
4905 </tp:Certificate>
4906 <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
4907     <tp:TrustAnchors>
4908         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1" />
4909         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2" />
4910         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4" />
4911     </tp:TrustAnchors>
4912 </tp:SecurityDetails>
4913 <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
4914     <tp:TrustAnchors>
4915         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3" />
4916         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5" />
4917     </tp:TrustAnchors>

```

```

4918     </tp:SecurityDetails>
4919     <tp:DeliveryChannel
4920         tp:channelId="asyncChannelA1"
4921         tp:transportId="transportA1"
4922         tp:docExchangeId="docExchangeA1">
4923         <tp:MessagingCharacteristics
4924             tp:syncReplyMode="none"
4925             tp:ackRequested="always"
4926             tp:ackSignatureRequested="always"
4927             tp:duplicateElimination="always"/>
4928     </tp:DeliveryChannel>
4929     <tp:DeliveryChannel
4930         tp:channelId="syncChannelA1"
4931         tp:transportId="transportA2"
4932         tp:docExchangeId="docExchangeA1">
4933         <tp:MessagingCharacteristics
4934             tp:syncReplyMode="signalsAndResponse"
4935             tp:ackRequested="always"
4936             tp:ackSignatureRequested="always"
4937             tp:duplicateElimination="always"/>
4938     </tp:DeliveryChannel>
4939     <tp:Transport tp:transportId="transportA1">
4940         <tp:TransportSender>
4941             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4942             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4943             <tp:TransportClientSecurity>
4944                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4945                 <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
4946                 <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4947             </tp:TransportClientSecurity>
4948         </tp:TransportSender>
4949         <tp:TransportReceiver>
4950             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4951             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4952             <tp:Endpoint
4953                 tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/async"
4954                 tp:type="allPurpose"/>
4955             <tp:TransportServerSecurity>
4956                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4957                 <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
4958                 <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4959             </tp:TransportServerSecurity>
4960         </tp:TransportReceiver>
4961     </tp:Transport>
4962     <tp:Transport tp:transportId="transportA2">
4963         <tp:TransportSender>
4964             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4965             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4966             <tp:TransportClientSecurity>
4967                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4968                 <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
4969                 <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4970             </tp:TransportClientSecurity>
4971         </tp:TransportSender>
4972         <tp:TransportReceiver>
4973             <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4974             <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4975             <tp:Endpoint
4976                 tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
4977                 tp:type="allPurpose"/>
4978             <tp:TransportServerSecurity>
4979                 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4980                 <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
4981                 <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4982             </tp:TransportServerSecurity>
4983         </tp:TransportReceiver>
4984     </tp:Transport>
4985     <tp:DocExchange tp:docExchangeId="docExchangeA1">
4986         <tp:ebXMLSenderBinding tp:version="2.0">
4987             <tp:ReliableMessaging>
4988                 <tp:Retries>3</tp:Retries>

```

```

4989         <tp:RetryInterval>PT2H</tp:RetryInterval>
4990         <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4991     </tp:ReliableMessaging>
4992     <tp:PersistDuration>P1D</tp:PersistDuration>
4993     <tp:SenderNonRepudiation>
4994
4995 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4996     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4997     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4998 sha1</tp:SignatureAlgorithm>
4999     <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert" />
5000 </tp:SenderNonRepudiation>
5001 <tp:SenderDigitalEnvelope>
5002     <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5003     <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5004     <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity" />
5005 </tp:SenderDigitalEnvelope>
5006 </tp:ebXMLSenderBinding>
5007 <tp:ebXMLReceiverBinding tp:version="2.0">
5008     <tp:ReliableMessaging>
5009         <tp:Retries>3</tp:Retries>
5010         <tp:RetryInterval>PT2H</tp:RetryInterval>
5011         <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
5012     </tp:ReliableMessaging>
5013     <tp:PersistDuration>P1D</tp:PersistDuration>
5014     <tp:ReceiverNonRepudiation>
5015
5016 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
5017     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
5018     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
5019 sha1</tp:SignatureAlgorithm>
5020     <tp:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity" />
5021 </tp:ReceiverNonRepudiation>
5022 <tp:ReceiverDigitalEnvelope>
5023     <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5024     <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5025     <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert" />
5026 </tp:ReceiverDigitalEnvelope>
5027 </tp:ebXMLReceiverBinding>
5028 </tp:DocExchange>
5029 </tp:PartyInfo>
5030 <!-- Party info for CompanyB -->
5031 <tp:PartyInfo
5032     tp:partyName="CompanyB"
5033     tp:defaultMshChannelId="asyncChannelB1"
5034     tp:defaultMshPackageId="CompanyB_MshSignalPackage">
5035 <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
5036 <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html" />
5037 <tp:CollaborationRole>
5038     <tp:ProcessSpecification
5039         tp:version="2.0"
5040         tp:name="PIP3A4RequestPurchaseOrder"
5041         xlink:type="simple"
5042         xlink:href="http://www.rosettanet.org/processes/3A4.xml"
5043         tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
5044 <tp:Role
5045     tp:name="Seller"
5046     xlink:type="simple"
5047     xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
5048 <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert" />
5049 <tp:ServiceBinding>
5050     <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
5051     <tp:CanSend>
5052         <tp:ThisPartyActionBinding
5053             tp:id="companyB_ABID1"
5054             tp:action="Purchase Order Confirmation Action"
5055             tp:packageId="CompanyB_ResponsePackage">
5056         <tp:BusinessTransactionCharacteristics
5057             tp:isNonRepudiationRequired="true"
5058             tp:isNonRepudiationReceiptRequired="true"
5059             tp:isSecureTransportRequired="true"

```

```

5060         tp:isConfidential="transient"
5061         tp:isAuthenticated="persistent"
5062         tp:isTamperProof="persistent"
5063         tp:isAuthorizationRequired="true"
5064         tp:timeToAcknowledgeReceipt="PT2H"
5065         tp:timeToPerform="P1D"/>
5066     <tp:ActionContext
5067         tp:binaryCollaboration="Request Purchase Order"
5068         tp:businessTransactionActivity="Request Purchase Order"
5069         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
5070     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5071 </tp:ThisPartyActionBinding>
5072 <tp:OtherPartyActionBinding>companyA_ABID3</tp:OtherPartyActionBinding>
5073 </tp:CanSend>
5074 <tp:CanSend>
5075     <tp:ThisPartyActionBinding
5076         tp:id="companyB_ABID2"
5077         tp:action="ReceiptAcknowledgement"
5078         tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
5079     <tp:BusinessTransactionCharacteristics
5080         tp:isNonRepudiationRequired="true"
5081         tp:isNonRepudiationReceiptRequired="true"
5082         tp:isSecureTransportRequired="true"
5083         tp:isConfidential="transient"
5084         tp:isAuthenticated="persistent"
5085         tp:isTamperProof="persistent"
5086         tp:isAuthorizationRequired="true"
5087         tp:timeToAcknowledgeReceipt="PT2H"
5088         tp:timeToPerform="P1D"/>
5089     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5090 </tp:ThisPartyActionBinding>
5091 <tp:OtherPartyActionBinding>companyA_ABID4</tp:OtherPartyActionBinding>
5092 </tp:CanSend>
5093 <tp:CanSend>
5094     <tp:ThisPartyActionBinding
5095         tp:id="companyB_ABID3"
5096         tp:action="Exception"
5097         tp:packageId="CompanyB_ExceptionPackage">
5098     <tp:BusinessTransactionCharacteristics
5099         tp:isNonRepudiationRequired="true"
5100         tp:isNonRepudiationReceiptRequired="true"
5101         tp:isSecureTransportRequired="true"
5102         tp:isConfidential="transient"
5103         tp:isAuthenticated="persistent"
5104         tp:isTamperProof="persistent"
5105         tp:isAuthorizationRequired="true"
5106         tp:timeToAcknowledgeReceipt="PT2H"
5107         tp:timeToPerform="P1D"/>
5108     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5109 </tp:ThisPartyActionBinding>
5110 <tp:OtherPartyActionBinding>companyA_ABID5</tp:OtherPartyActionBinding>
5111 </tp:CanSend>
5112 <tp:CanReceive>
5113     <tp:ThisPartyActionBinding
5114         tp:id="companyB_ABID4"
5115         tp:action="Purchase Order Request Action"
5116         tp:packageId="CompanyB_RequestPackage">
5117     <tp:BusinessTransactionCharacteristics
5118         tp:isNonRepudiationRequired="true"
5119         tp:isNonRepudiationReceiptRequired="true"
5120         tp:isSecureTransportRequired="true"
5121         tp:isConfidential="transient"
5122         tp:isAuthenticated="persistent"
5123         tp:isTamperProof="persistent"
5124         tp:isAuthorizationRequired="true"
5125         tp:timeToAcknowledgeReceipt="PT2H"
5126         tp:timeToPerform="P1D"/>
5127     <tp:ActionContext
5128         tp:binaryCollaboration="Request Purchase Order"
5129         tp:businessTransactionActivity="Request Purchase Order"
5130         tp:requestOrResponseAction="Purchase Order Request Action"/>

```

```

5131         <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5132     </tp:ThisPartyActionBinding>
5133     <tp:OtherPartyActionBinding>companyA_ABID1</tp:OtherPartyActionBinding>
5134 </tp:CanReceive>
5135 <tp:CanReceive>
5136     <tp:ThisPartyActionBinding
5137         tp:id="companyB_ABID5"
5138         tp:action="ReceiptAcknowledgment"
5139         tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
5140         <tp:BusinessTransactionCharacteristics
5141             tp:isNonRepudiationRequired="true"
5142             tp:isNonRepudiationReceiptRequired="true"
5143             tp:isSecureTransportRequired="true"
5144             tp:isConfidential="transient"
5145             tp:isAuthenticated="persistent"
5146             tp:isTamperProof="persistent"
5147             tp:isAuthorizationRequired="true"
5148             tp:timeToAcknowledgeReceipt="PT2H"
5149             tp:timeToPerform="P1D" />
5150         <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5151     </tp:ThisPartyActionBinding>
5152     <tp:OtherPartyActionBinding>companyA_ABID2</tp:OtherPartyActionBinding>
5153 </tp:CanReceive>
5154 <!-- The next binding uses a synchronous delivery channel -->
5155 <tp:CanReceive>
5156     <tp:ThisPartyActionBinding
5157         tp:id="companyB_ABID6"
5158         tp:action="Purchase Order Request Action"
5159         tp:packageId="CompanyB_RequestPackage">
5160         <tp:BusinessTransactionCharacteristics
5161             tp:isNonRepudiationRequired="true"
5162             tp:isNonRepudiationReceiptRequired="true"
5163             tp:isSecureTransportRequired="true"
5164             tp:isConfidential="transient"
5165             tp:isAuthenticated="persistent"
5166             tp:isTamperProof="persistent"
5167             tp:isAuthorizationRequired="true"
5168             tp:timeToAcknowledgeReceipt="PT5M" tp:timeToPerform="PT5M" />
5169         <tp>ActionContext
5170             tp:binaryCollaboration="Request Purchase Order"
5171             tp:businessTransactionActivity="Request Purchase Order"
5172             tp:requestOrResponseAction="Purchase Order Request Action" />
5173         <tp:ChannelId>syncChannelB1</tp:ChannelId>
5174     </tp:ThisPartyActionBinding>
5175     <tp:OtherPartyActionBinding>companyA_ABID6</tp:OtherPartyActionBinding>
5176 <tp:CanSend>
5177     <tp:ThisPartyActionBinding
5178         tp:id="companyB_ABID7"
5179         tp:action="Purchase Order Confirmation Action"
5180         tp:packageId="CompanyB_SyncReplyPackage">
5181         <tp:BusinessTransactionCharacteristics
5182             tp:isNonRepudiationRequired="true"
5183             tp:isNonRepudiationReceiptRequired="true"
5184             tp:isSecureTransportRequired="true"
5185             tp:isConfidential="transient"
5186             tp:isAuthenticated="persistent"
5187             tp:isTamperProof="persistent"
5188             tp:isAuthorizationRequired="true"
5189             tp:timeToAcknowledgeReceipt="PT5M"
5190             tp:timeToPerform="PT5M" />
5191         <tp>ActionContext
5192             tp:binaryCollaboration="Request Purchase Order"
5193             tp:businessTransactionActivity="Request Purchase Order"
5194             tp:requestOrResponseAction="Purchase Order Confirmation Action" />
5195         <tp:ChannelId>syncChannelB1</tp:ChannelId>
5196     </tp:ThisPartyActionBinding>
5197     <tp:OtherPartyActionBinding>companyA_ABID7</tp:OtherPartyActionBinding>
5198 </tp:CanSend>
5199 <tp:CanSend>
5200     <tp:ThisPartyActionBinding
5201         tp:id="companyB_ABID8"

```

```
5202         tp:action="Exception"
5203         tp:packageId="CompanyB_ExceptionPackage">
5204         <tp:BusinessTransactionCharacteristics
5205             tp:isNonRepudiationRequired="true"
5206             tp:isNonRepudiationReceiptRequired="true"
5207             tp:isSecureTransportRequired="true"
5208             tp:isConfidential="transient"
5209             tp:isAuthenticated="persistent"
5210             tp:isTamperProof="persistent"
5211             tp:isAuthorizationRequired="true"
5212             tp:timeToAcknowledgeReceipt="PT5M"
5213             tp:timeToPerform="PT5M" />
5214         <tp:ChannelId>syncChannelB1</tp:ChannelId>
5215     </tp:ThisPartyActionBinding>
5216     <tp:OtherPartyActionBinding>companyA_ABID8</tp:OtherPartyActionBinding>
5217 </tp:CanSend>
5218 </tp:CanReceive>
5219 </tp:ServiceBinding>
5220 </tp:CollaborationRole>
5221 <!-- Certificates used by the "Seller" company -->
5222 <tp:Certificate tp:certId="CompanyB_AppCert">
5223     <ds:KeyInfo>
5224         <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
5225     </ds:KeyInfo>
5226 </tp:Certificate>
5227 <tp:Certificate tp:certId="CompanyB_SigningCert">
5228     <ds:KeyInfo>
5229         <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
5230     </ds:KeyInfo>
5231 </tp:Certificate>
5232 <tp:Certificate tp:certId="CompanyB_EncryptionCert">
5233     <ds:KeyInfo>
5234         <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
5235     </ds:KeyInfo>
5236 </tp:Certificate>
5237 <tp:Certificate tp:certId="CompanyB_ServerCert">
5238     <ds:KeyInfo>
5239         <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
5240     </ds:KeyInfo>
5241 </tp:Certificate>
5242 <tp:Certificate tp:certId="CompanyB_ClientCert">
5243     <ds:KeyInfo>
5244         <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
5245     </ds:KeyInfo>
5246 </tp:Certificate>
5247 <tp:Certificate tp:certId="TrustedRootCertB4">
5248     <ds:KeyInfo>
5249         <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
5250     </ds:KeyInfo>
5251 </tp:Certificate>
5252 <tp:Certificate tp:certId="TrustedRootCertB5">
5253     <ds:KeyInfo>
5254         <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
5255     </ds:KeyInfo>
5256 </tp:Certificate>
5257 <tp:Certificate tp:certId="TrustedRootCertB6">
5258     <ds:KeyInfo>
5259         <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
5260     </ds:KeyInfo>
5261 </tp:Certificate>
5262 <tp:Certificate tp:certId="TrustedRootCertB7">
5263     <ds:KeyInfo>
5264         <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
5265     </ds:KeyInfo>
5266 </tp:Certificate>
5267 <tp:Certificate tp:certId="TrustedRootCertB8">
5268     <ds:KeyInfo>
5269         <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
5270     </ds:KeyInfo>
5271 </tp:Certificate>
5272 <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
```

```

5273     <tp:TrustAnchors>
5274         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5" />
5275         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6" />
5276         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4" />
5277     </tp:TrustAnchors>
5278 </tp:SecurityDetails>
5279 <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
5280     <tp:TrustAnchors>
5281         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8" />
5282         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7" />
5283     </tp:TrustAnchors>
5284 </tp:SecurityDetails>
5285 <!-- An asynchronous delivery channel -->
5286 <tp:DeliveryChannel
5287     tp:channelId="asyncChannelB1"
5288     tp:transportId="transportB1"
5289     tp:docExchangeId="docExchangeB1">
5290     <tp:MessagingCharacteristics
5291         tp:syncReplyMode="none"
5292         tp:ackRequested="always"
5293         tp:ackSignatureRequested="always"
5294         tp:duplicateElimination="always" />
5295 </tp:DeliveryChannel>
5296 <!-- A synchronous delivery channel -->
5297 <tp:DeliveryChannel
5298     tp:channelId="syncChannelB1"
5299     tp:transportId="transportB2"
5300     tp:docExchangeId="docExchangeB1">
5301     <tp:MessagingCharacteristics
5302         tp:syncReplyMode="signalsAndResponse"
5303         tp:ackRequested="always"
5304         tp:ackSignatureRequested="always"
5305         tp:duplicateElimination="always" />
5306 </tp:DeliveryChannel>
5307 <tp:Transport tp:transportId="transportB1">
5308     <tp:TransportSender>
5309         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5310         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5311         <tp:TransportClientSecurity>
5312             <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5313             <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert" />
5314             <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity" />
5315         </tp:TransportClientSecurity>
5316     </tp:TransportSender>
5317     <tp:TransportReceiver>
5318         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5319         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5320         <tp:Endpoint
5321             tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
5322             tp:type="allPurpose" />
5323         <tp:TransportServerSecurity>
5324             <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5325             <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert" />
5326             <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity" />
5327         </tp:TransportServerSecurity>
5328     </tp:TransportReceiver>
5329 </tp:Transport>
5330 <tp:Transport tp:transportId="transportB2">
5331     <tp:TransportSender>
5332         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5333         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5334         <tp:TransportClientSecurity>
5335             <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5336             <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert" />
5337             <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity" />
5338         </tp:TransportClientSecurity>
5339     </tp:TransportSender>
5340     <tp:TransportReceiver>
5341         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
5342         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
5343         <tp:Endpoint

```



```

5344         tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
5345         tp:type="allPurpose"/>
5346     <tp:TransportServerSecurity>
5347         <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
5348         <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
5349         <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
5350     </tp:TransportServerSecurity>
5351 </tp:TransportReceiver>
5352 </tp:Transport>
5353 <tp:DocExchange tp:docExchangeId="docExchangeB1">
5354     <tp:ebXMLSenderBinding tp:version="2.0">
5355         <tp:ReliableMessaging>
5356             <tp:Retries>3</tp:Retries>
5357             <tp:RetryInterval>PT2H</tp:RetryInterval>
5358             <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
5359         </tp:ReliableMessaging>
5360         <tp:PersistDuration>P1D</tp:PersistDuration>
5361         <tp:SenderNonRepudiation>
5362
5363 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
5364         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
5365         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
5366 sha1</tp:SignatureAlgorithm>
5367         <tp:SigningCertificateRef tp:certId="CompanyB_SigningCert"/>
5368     </tp:SenderNonRepudiation>
5369     <tp:SenderDigitalEnvelope>
5370         <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5371         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5372         <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
5373     </tp:SenderDigitalEnvelope>
5374 </tp:ebXMLSenderBinding>
5375 <tp:ebXMLReceiverBinding tp:version="2.0">
5376     <tp:ReliableMessaging>
5377         <tp:Retries>3</tp:Retries>
5378         <tp:RetryInterval>PT2H</tp:RetryInterval>
5379         <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
5380     </tp:ReliableMessaging>
5381     <tp:PersistDuration>P1D</tp:PersistDuration>
5382     <tp:ReceiverNonRepudiation>
5383
5384 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
5385         <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
5386         <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
5387 sha1</tp:SignatureAlgorithm>
5388         <tp:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
5389     </tp:ReceiverNonRepudiation>
5390     <tp:ReceiverDigitalEnvelope>
5391         <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5392         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5393         <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert"/>
5394     </tp:ReceiverDigitalEnvelope>
5395 </tp:ebXMLReceiverBinding>
5396 </tp:DocExchange>
5397 </tp:PartyInfo>
5398 <!-- SimplePart corresponding to the SOAP Envelope -->
5399 <tp:SimplePart
5400     tp:id="CompanyA_MsgHdr"
5401     tp:mimetype="text/xml">
5402     <tp:NamespaceSupported
5403         tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5404         tp:version="2.0">
5405         http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5406     </tp:NamespaceSupported>
5407 </tp:SimplePart>
5408 <tp:SimplePart
5409     tp:id="CompanyB_MsgHdr"
5410     tp:mimetype="text/xml">
5411     <tp:NamespaceSupported
5412         tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5413         tp:version="2.0">
5414         http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd

```

```
5415 </tp:NamespaceSupported>
5416 </tp:SimplePart>
5417 <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
5418 <tp:SimplePart
5419   tp:id="CompanyA_ReceiptAcknowledgment"
5420   tp:mimetype="application/xml">
5421   <tp:NamespaceSupported
5422     tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
5423     tp:version="2.0">http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
5424   </tp:NamespaceSupported>
5425 </tp:SimplePart>
5426 <tp:SimplePart
5427   tp:id="CompanyB_ReceiptAcknowledgment"
5428   tp:mimetype="application/xml">
5429   <tp:NamespaceSupported
5430     tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
5431     tp:version="2.0">
5432     http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
5433   </tp:NamespaceSupported>
5434 </tp:SimplePart>
5435 <!-- SimplePart corresponding to an Exception business signal -->
5436 <tp:SimplePart
5437   tp:id="CompanyA_Exception"
5438   tp:mimetype="application/xml">
5439   <tp:NamespaceSupported
5440     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5441     tp:version="2.0">
5442     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5443   </tp:NamespaceSupported>
5444 </tp:SimplePart>
5445 <tp:SimplePart
5446   tp:id="CompanyB_Exception"
5447   tp:mimetype="application/xml">
5448   <tp:NamespaceSupported
5449     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5450     tp:version="2.0">
5451     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5452   </tp:NamespaceSupported>
5453 </tp:SimplePart>
5454 <!-- SimplePart corresponding to a request action -->
5455 <tp:SimplePart
5456   tp:id="CompanyA_Request"
5457   tp:mimetype="application/xml">
5458   <tp:NamespaceSupported
5459     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
5460     tp:version="1.0">
5461     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
5462   </tp:NamespaceSupported>
5463 </tp:SimplePart>
5464 <tp:SimplePart
5465   tp:id="CompanyB_Request"
5466   tp:mimetype="application/xml">
5467   <tp:NamespaceSupported
5468     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
5469     tp:version="1.0">
5470     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
5471   </tp:NamespaceSupported>
5472 </tp:SimplePart>
5473 <!-- SimplePart corresponding to a response action -->
5474 <tp:SimplePart
5475   tp:id="CompanyA_Response"
5476   tp:mimetype="application/xml">
5477   <tp:NamespaceSupported
5478     tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
5479     tp:version="1.0">
5480     http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
5481   </tp:NamespaceSupported>
5482 </tp:SimplePart>
5483 <tp:SimplePart
5484   tp:id="CompanyB_Response"
5485   tp:mimetype="application/xml">
```

```
5486     <tp:NamespaceSupported
5487         tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
5488         tp:version="1.0">
5489         http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
5490     </tp:NamespaceSupported>
5491 </tp:SimplePart>
5492 <!-- An ebXML message with a SOAP Envelope only -->
5493 <tp:Packaging
5494     tp:id="CompanyA_MshSignalPackage">
5495     <tp:ProcessingCapabilities
5496         tp:parse="true"
5497         tp:generate="true"/>
5498     <tp:CompositeList>
5499         <tp:Composite
5500             tp:id="CompanyA_MshSignal"
5501             tp:mimetype="multipart/related"
5502             tp:mimeparameters="type=text/xml">
5503             <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5504         </tp:Composite>
5505     </tp:CompositeList>
5506 </tp:Packaging>
5507 <tp:Packaging
5508     tp:id="CompanyB_MshSignalPackage">
5509     <tp:ProcessingCapabilities
5510         tp:parse="true"
5511         tp:generate="true"/>
5512     <tp:CompositeList>
5513         <tp:Composite
5514             tp:id="CompanyB_MshSignal"
5515             tp:mimetype="multipart/related"
5516             tp:mimeparameters="type=text/xml">
5517             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5518         </tp:Composite>
5519     </tp:CompositeList>
5520 </tp:Packaging>
5521 <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
5522 <tp:Packaging tp:id="CompanyA_RequestPackage">
5523     <tp:ProcessingCapabilities
5524         tp:parse="true"
5525         tp:generate="true"/>
5526     <tp:CompositeList>
5527         <tp:Composite
5528             tp:id="CompanyA_RequestMsg"
5529             tp:mimetype="multipart/related"
5530             tp:mimeparameters="type=text/xml">
5531             <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5532             <tp:Constituent tp:idref="CompanyA_Request"/>
5533         </tp:Composite>
5534     </tp:CompositeList>
5535 </tp:Packaging>
5536 <tp:Packaging tp:id="CompanyB_RequestPackage">
5537     <tp:ProcessingCapabilities
5538         tp:parse="true"
5539         tp:generate="true"/>
5540     <tp:CompositeList>
5541         <tp:Composite
5542             tp:id="CompanyB_RequestMsg"
5543             tp:mimetype="multipart/related"
5544             tp:mimeparameters="type=text/xml">
5545             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5546             <tp:Constituent tp:idref="CompanyB_Request"/>
5547         </tp:Composite>
5548     </tp:CompositeList>
5549 </tp:Packaging>
5550 <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
5551 <tp:Packaging tp:id="CompanyA_ResponsePackage">
5552     <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>
5553     <tp:CompositeList>
5554         <tp:Composite
5555             tp:id="CompanyA_ResponseMsg"
5556             tp:mimetype="multipart/related"
```

```

5557         tp:mimeparameters="type=text/xml">
5558         <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5559         <tp:Constituent tp:idref="CompanyA_Response"/>
5560     </tp:Composite>
5561 </tp:CompositeList>
5562 </tp:Packaging>
5563 <tp:Packaging tp:id="CompanyB_ResponsePackage">
5564     <tp:ProcessingCapabilities
5565         tp:parse="true"
5566         tp:generate="true"/>
5567     <tp:CompositeList>
5568         <tp:Composite
5569             tp:id="CompanyB_ResponseMsg"
5570             tp:mimetype="multipart/related"
5571             tp:mimeparameters="type=text/xml">
5572             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5573             <tp:Constituent tp:idref="CompanyB_Response"/>
5574         </tp:Composite>
5575     </tp:CompositeList>
5576 </tp:Packaging>
5577 <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
5578 <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
5579     <tp:ProcessingCapabilities
5580         tp:parse="true"
5581         tp:generate="true"/>
5582     <tp:CompositeList>
5583         <tp:Composite
5584             tp:id="CompanyA_ReceiptAcknowledgmentMsg"
5585             tp:mimetype="multipart/related"
5586             tp:mimeparameters="type=text/xml">
5587             <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5588             <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
5589         </tp:Composite>
5590     </tp:CompositeList>
5591 </tp:Packaging>
5592 <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
5593     <tp:ProcessingCapabilities
5594         tp:parse="true"
5595         tp:generate="true"/>
5596     <tp:CompositeList>
5597         <tp:Composite
5598             tp:id="CompanyB_ReceiptAcknowledgmentMsg"
5599             tp:mimetype="multipart/related"
5600             tp:mimeparameters="type=text/xml">
5601             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5602             <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
5603         </tp:Composite>
5604     </tp:CompositeList>
5605 </tp:Packaging>
5606 <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
5607 <tp:Packaging tp:id="CompanyA_ExceptionPackage">
5608     <tp:ProcessingCapabilities
5609         tp:parse="true"
5610         tp:generate="true"/>
5611     <tp:CompositeList>
5612         <tp:Composite
5613             tp:id="CompanyA_ExceptionMsg"
5614             tp:mimetype="multipart/related"
5615             tp:mimeparameters="type=text/xml">
5616             <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5617             <tp:Constituent tp:idref="CompanyA_Exception"/>
5618         </tp:Composite>
5619     </tp:CompositeList>
5620 </tp:Packaging>
5621 <tp:Packaging tp:id="CompanyB_ExceptionPackage">
5622     <tp:ProcessingCapabilities
5623         tp:parse="true"
5624         tp:generate="true"/>
5625     <tp:CompositeList>
5626         <tp:Composite
5627             tp:id="CompanyB_ExceptionMsg"

```

```
5628         tp:mimetype="multipart/related"
5629         tp:mimeparameters="type=text/xml">
5630         <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5631         <tp:Constituent tp:idref="CompanyB_Exception"/>
5632     </tp:Composite>
5633 </tp:CompositeList>
5634 </tp:Packaging>
5635 <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
5636      or an ebXML message with an Exception signal -->
5637 <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
5638     <tp:ProcessingCapabilities
5639         tp:parse="true"
5640         tp:generate="true"/>
5641     <tp:CompositeList>
5642         <tp:Composite
5643             tp:id="CompanyA_SignalAndResponseMsg"
5644             tp:mimetype="multipart/related"
5645             tp:mimeparameters="type=text/xml">
5646             <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5647             <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
5648             <tp:Constituent tp:idref="CompanyA_Response"/>
5649             </tp:Composite>
5650         </tp:CompositeList>
5651     </tp:Packaging>
5652 <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
5653     <tp:ProcessingCapabilities
5654         tp:parse="true"
5655         tp:generate="true"/>
5656     <tp:CompositeList>
5657         <tp:Composite
5658             tp:id="CompanyB_SignalAndResponseMsg"
5659             tp:mimetype="multipart/related"
5660             tp:mimeparameters="type=text/xml">
5661             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5662             <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
5663             <tp:Constituent tp:idref="CompanyB_Response"/>
5664             </tp:Composite>
5665         </tp:CompositeList>
5666     </tp:Packaging>
5667     <tp:Comment xml:lang="en-US">buy/sell agreement between CompanyA.com and
5668     CompanyB.com</tp:Comment>
5669 </tp:CollaborationProtocolAgreement>
5670
```

5671 **Appendix C Business Process Specification Corresponding**  
 5672 **to Complete CPP and CPA Definition (Non-Normative)**

5673 This Business Process Specification referenced by the CPPs and CPA in Appendix A and  
 5674 Appendix B are reproduced here. This document is available as an ASCII file at:

5675 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-bpss-example-017.xml>  
 5676

5677 The schema to which this instance document conforms is available as an ASCII file at:

5678 <http://www.oasis-open.org/committees/ebxml-cppa/schema/ebBPSS1.03.xsd>  
 5679

```

5680 <?xml version="1.0" encoding="UTF-8"?>
5681 <ProcessSpecification
5682   xmlns="http://www.ebxml.org/BusinessProcess"
5683   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5684   xsi:schemaLocation="http://www.ebxml.org/BusinessProcess ebBPSS1.03.xsd"
5685   name="PIP3A4RequestPurchaseOrder"
5686   uuid="urn:icann:rosettnet.org:bpid:3A4$2.0"
5687   version="R02.00">
5688   <Documentation>
5689     This PIP enables a buyer to issue a purchase order and obtain a quick response
5690     from the provider that acknowledges which of the purchase order product line
5691     items are accepted, rejected, or pending.
5692   </Documentation>
5693   <!--Purchase order Request Document-->
5694   <BusinessDocument
5695     name="Purchase Order Request"
5696     nameID="Pip3A4PurchaseOrderRequest"
5697     specificationLocation="PurchaseOrderRequest.xsd">
5698     <Documentation>
5699       The document is an XSD file that specifies the rules for creating the XML
5700       document for the business action of requesting a purchase order.
5701     </Documentation>
5702   </BusinessDocument>
5703   <BusinessDocument
5704     name="Purchase Order Confirmation"
5705     nameID="Pip3A4PurchaseOrderConfirmation"
5706     specificationLocation="PurchaseOrderConfirmation.xsd">
5707     <Documentation>
5708       The document is an XSD file that specifies the rules for creating the XML
5709       document for the business action of making a purchase order confirmation.
5710     </Documentation>
5711   </BusinessDocument>
5712   <BusinessTransaction
5713     name="Request Purchase Order"
5714     nameID="RequestPurchaseOrder_BT">
5715     <RequestingBusinessActivity
5716       name="Purchase Order Request Action"
5717       nameID="PurchaseOrderRequestAction"
5718       isAuthorizationRequired="true"
5719       isIntelligibleCheckRequired="true"
5720       isNonRepudiationReceiptRequired="true"
5721       isNonRepudiationRequired="true"
5722       timeToAcknowledgeReceipt="POY0M0DT2H0M0S">
5723       <DocumentEnvelope
5724         businessDocument="Purchase Order Request"
5725         businessDocumentIDRef="Pip3A4PurchaseOrderRequest"
5726         isAuthenticated="persistent"
5727         isConfidential="transient"
5728         isTamperProof="persistent"/>
5729     </RequestingBusinessActivity>
5730     <RespondingBusinessActivity
5731       name="Purchase Order Confirmation Action"
5732       nameID="PurchaseOrderConfirmationAction"
5733       isAuthorizationRequired="true"
  
```

```
5734     isIntelligibleCheckRequired="true"
5735     isNonRepudiationReceiptRequired="false"
5736     isNonRepudiationRequired="true"
5737     timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
5738     <DocumentEnvelope
5739         businessDocument="Purchase Order Confirmation"
5740         businessDocumentIDRef="Pip3A4PurchaseOrderConfirmation"
5741         isAuthenticated="persistent"
5742         isConfidential="transient"
5743         isPositiveResponse="true"
5744         isTamperProof="persistent" />
5745     </RespondingBusinessActivity>
5746 </BusinessTransaction>
5747 <BinaryCollaboration
5748     name="Request Purchase Order"
5749     nameID="RequestPurchaseOrder_BC">
5750     <InitiatingRole
5751         name="Buyer"
5752         nameID="BuyerId" />
5753     <RespondingRole
5754         name="Seller"
5755         nameID="SellerId" />
5756     <Start toBusinessState="Request Purchase Order" />
5757     <BusinessTransactionActivity
5758         name="Request Purchase Order"
5759         nameID="RequestPurchaseOrder_BTA"
5760         businessTransaction="Request Purchase Order"
5761         businessTransactionIDRef="RequestPurchaseOrder_BT"
5762         fromAuthorizedRole="Buyer" fromAuthorizedRoleIDRef="BuyerId"
5763         toAuthorizedRole="Seller" toAuthorizedRoleIDRef="SellerId"
5764         isLegallyBinding="true"
5765         timeToPerform="P0Y0M0DT24H0M0S"
5766         isConcurrent="false" />
5767     <Transition
5768         fromBusinessState="Request Purchase Order"
5769         toBusinessState="Request Purchase Order" />
5770     <Success
5771         fromBusinessState="Request Purchase Order"
5772         conditionGuard="Success" />
5773     <Failure
5774         fromBusinessState="Request Purchase Order"
5775         conditionGuard="BusinessFailure" />
5776 </BinaryCollaboration>
5777 </ProcessSpecification>
5778
```

5779 Appendix D W3C XML Schema Document Corresponding to  
5780 Complete CPP and CPA Definition (Normative)

5781 This XML Schema document is available as an ASCII file at:

5782 <http://www.oasis-open.org/committees/ebxml-cppa/schema/draft-cpp-cpa-017.xsd>

```
5783
5784 <?xml version="1.0" encoding="UTF-8"?>
5785 <!-- Some parsers may require explicit declaration of
5786 'xmlns:xml="http://www.w3.org/XML/1998/namespace"'.
5787 In that case, a copy of this schema augmented with the above declaration should be cached
5788 and used
5789 for the purpose of schema validation for CPPs and CPAs. -->
5790 <schema
5791   targetNamespace="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
5792   xmlns:tns="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
5793   xmlns="http://www.w3.org/2001/XMLSchema"
5794   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5795   xmlns:xlink="http://www.w3.org/1999/xlink"
5796   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
5797   elementFormDefault="qualified"
5798   attributeFormDefault="qualified" version="017">
5799   <import
5800     namespace="http://www.w3.org/1999/xlink"
5801     schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/xlink.xsd"/>
5802   <import
5803     namespace="http://www.w3.org/2000/09/xmldsig#"
5804     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
5805   <import
5806     namespace="http://www.w3.org/XML/1998/namespace"
5807     schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
5808   <attributeGroup name="pkg.grp">
5809     <attribute ref="tns:id" use="required"/>
5810     <attribute name="mimetype" type="tns:non-empty-string" use="required"/>
5811     <attribute name="mimeparameters" type="tns:non-empty-string"/>
5812   </attributeGroup>
5813   <attributeGroup name="xlink.grp">
5814     <attribute ref="xlink:type" fixed="simple"/>
5815     <attribute ref="xlink:href" use="required"/>
5816   </attributeGroup>
5817   <element name="CollaborationProtocolAgreement">
5818     <complexType>
5819       <sequence>
5820         <element ref="tns:Status"/>
5821         <element ref="tns:Start"/>
5822         <element ref="tns:End"/>
5823         <element ref="tns:ConversationConstraints" minOccurs="0"/>
5824         <element ref="tns:PartyInfo" minOccurs="2" maxOccurs="2"/>
5825         <element ref="tns:SimplePart" maxOccurs="unbounded"/>
5826         <element ref="tns:Packaging" maxOccurs="unbounded"/>
5827         <element ref="tns:Signature" minOccurs="0"/>
5828         <element ref="tns:Comment" minOccurs="0" maxOccurs="unbounded"/>
5829       </sequence>
5830       <attribute name="cpaid" type="tns:non-empty-string" use="required"/>
5831       <attribute ref="tns:version" use="required"/>
5832     </complexType>
5833   </element>
5834   <element name="Signature">
5835     <complexType>
5836       <sequence>
5837         <element ref="ds:Signature" maxOccurs="3"/>
5838       </sequence>
5839     </complexType>
5840   </element>
5841   <element name="CollaborationProtocolProfile">
5842     <complexType>
5843       <sequence>
```



```

5844     <element ref="tns:PartyInfo" maxOccurs="unbounded" />
5845     <element ref="tns:SimplePart" maxOccurs="unbounded" />
5846     <element ref="tns:Packaging" maxOccurs="unbounded" />
5847     <element ref="tns:Signature" minOccurs="0" />
5848     <element ref="tns:Comment" minOccurs="0" maxOccurs="unbounded" />
5849   </sequence>
5850   <attribute name="cppid" type="tns:non-empty-string" use="required" />
5851   <attribute ref="tns:version" use="required" />
5852 </complexType>
5853 </element>
5854 <element name="ProcessSpecification">
5855   <complexType>
5856     <sequence>
5857       <element ref="ds:Reference" minOccurs="0" maxOccurs="unbounded" />
5858     </sequence>
5859     <attribute name="name" type="tns:non-empty-string" use="required" />
5860     <attribute ref="tns:version" use="required" />
5861     <attributeGroup ref="tns:xlink.grp" />
5862     <attribute name="uuid" type="anyURI" />
5863   </complexType>
5864 </element>
5865 <element name="Service" type="tns:service.type" />
5866 <element name="Protocol" type="tns:protocol.type" />
5867 <element name="SendingProtocol" type="tns:protocol.type" />
5868 <element name="ReceivingProtocol" type="tns:protocol.type" />
5869 <element name="OverrideMshActionBinding">
5870   <complexType>
5871     <attribute name="action" type="tns:non-empty-string" use="required" />
5872     <attribute name="channelId" type="IDREF" use="required" />
5873   </complexType>
5874 </element>
5875 <element name="ChannelId" type="IDREF" />
5876 <complexType name="ActionBinding.type">
5877   <sequence>
5878     <element ref="tns:BusinessTransactionCharacteristics" />
5879     <element ref="tns:ActionContext" minOccurs="0" />
5880     <element ref="tns:ChannelId" maxOccurs="unbounded" />
5881     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
5882   </sequence>
5883   <attribute name="id" type="ID" use="required" />
5884   <attribute name="action" type="tns:non-empty-string" use="required" />
5885   <attribute name="packageId" type="IDREF" use="required" />
5886   <attribute ref="xlink:href" use="optional" />
5887   <attribute ref="xlink:type" fixed="simple" />
5888 </complexType>
5889 <element name="ActionContext">
5890   <complexType>
5891     <sequence>
5892       <element ref="tns:CollaborationActivity" minOccurs="0" />
5893       <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
5894     </sequence>
5895     <attribute name="binaryCollaboration" type="tns:non-empty-string" use="required" />
5896     <attribute name="businessTransactionActivity" type="tns:non-empty-string" use="required" />
5897     <attribute name="requestOrResponseAction" type="tns:non-empty-string" use="required" />
5898   </complexType>
5899 </element>
5900 <element name="CollaborationActivity">
5901   <complexType>
5902     <sequence>
5903       <element ref="tns:CollaborationActivity" minOccurs="0" />
5904     </sequence>
5905     <attribute name="name" type="tns:non-empty-string" />
5906   </complexType>
5907 </element>
5908 <element name="CollaborationRole">
5909   <complexType>
5910     <sequence>
5911       <element ref="tns:ProcessSpecification" />
5912       <element ref="tns:Role" />
5913       <element name="ApplicationCertificateRef" type="tns:CertificateRef.type" minOccurs="0"
5914 maxOccurs="unbounded" />

```

```

5915         <element name="ApplicationSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
5916 minOccurs="0" />
5917         <element ref="tns:ServiceBinding" />
5918     </sequence>
5919 </complexType>
5920 </element>
5921 <element name="PartyInfo">
5922     <complexType>
5923         <sequence>
5924             <element ref="tns:PartyId" maxOccurs="unbounded" />
5925             <element ref="tns:PartyRef" maxOccurs="unbounded" />
5926             <element ref="tns:CollaborationRole" maxOccurs="unbounded" />
5927             <element ref="tns:Certificate" maxOccurs="unbounded" />
5928             <element ref="tns:SecurityDetails" maxOccurs="unbounded" />
5929             <element ref="tns:DeliveryChannel" maxOccurs="unbounded" />
5930             <element ref="tns:Transport" maxOccurs="unbounded" />
5931             <element ref="tns:DocExchange" maxOccurs="unbounded" />
5932             <element ref="tns:OverrideMshActionBinding" minOccurs="0" maxOccurs="unbounded" />
5933         </sequence>
5934         <attribute name="partyName" type="tns:non-empty-string" use="required" />
5935         <attribute name="defaultMshChannelId" type="IDREF" use="required" />
5936         <attribute name="defaultMshPackageId" type="IDREF" use="required" />
5937     </complexType>
5938 </element>
5939 <element name="PartyId">
5940     <complexType>
5941         <simpleContent>
5942             <extension base="tns:non-empty-string">
5943                 <attribute name="type" type="anyURI" />
5944             </extension>
5945         </simpleContent>
5946     </complexType>
5947 </element>
5948 <element name="PartyRef">
5949     <complexType>
5950         <sequence>
5951             </sequence>
5952         <attributeGroup ref="tns:xlink.grp" />
5953         <attribute name="type" type="anyURI" />
5954         <attribute name="schemaLocation" type="anyURI" />
5955     </complexType>
5956 </element>
5957 <element name="DeliveryChannel">
5958     <complexType>
5959         <sequence>
5960             <element ref="tns:MessagingCharacteristics" />
5961         </sequence>
5962         <attribute name="channelId" type="ID" use="required" />
5963         <attribute name="transportId" type="IDREF" use="required" />
5964         <attribute name="docExchangeId" type="IDREF" use="required" />
5965     </complexType>
5966 </element>
5967 <element name="Transport">
5968     <complexType>
5969         <sequence>
5970             <element ref="tns:TransportSender" minOccurs="0" />
5971             <element ref="tns:TransportReceiver" minOccurs="0" />
5972         </sequence>
5973         <attribute name="transportId" type="ID" use="required" />
5974     </complexType>
5975 </element>
5976 <element name="AccessAuthentication" type="tns:accessAuthentication.type" />
5977 <element name="TransportSender">
5978     <complexType>
5979         <sequence>
5980             <element name="TransportProtocol" type="tns:protocol.type" />
5981             <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded" />
5982             <element ref="tns:TransportClientSecurity" minOccurs="0" />
5983         </sequence>
5984     </complexType>
5985 </element>

```

```

5986 <element name="TransportReceiver">
5987 <complexType>
5988 <sequence>
5989 <element name="TransportProtocol" type="tns:protocol.type"/>
5990 <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded"/>
5991 <element ref="tns:Endpoint" maxOccurs="unbounded"/>
5992 <element ref="tns:TransportServerSecurity" minOccurs="0"/>
5993 </sequence>
5994 </complexType>
5995 </element>
5996 <element name="Endpoint">
5997 <complexType>
5998 <attribute name="uri" type="anyURI" use="required"/>
5999 <attribute name="type" type="tns:endpointType.type" default="allPurpose"/>
6000 </complexType>
6001 </element>
6002 <element name="TransportClientSecurity">
6003 <complexType>
6004 <sequence>
6005 <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
6006 <element name="ClientCertificateRef" type="tns:CertificateRef.type" minOccurs="0"/>
6007 <element name="ServerSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6008 minOccurs="0"/>
6009 <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
6010 </sequence>
6011 </complexType>
6012 </element>
6013 <element name="TransportServerSecurity">
6014 <complexType>
6015 <sequence>
6016 <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
6017 <element name="ServerCertificateRef" type="tns:CertificateRef.type"/>
6018 <element name="ClientSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6019 minOccurs="0"/>
6020 <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
6021 </sequence>
6022 </complexType>
6023 </element>
6024 <element name="Certificate">
6025 <complexType>
6026 <sequence>
6027 <element ref="ds:KeyInfo"/>
6028 </sequence>
6029 <attribute name="certId" type="ID" use="required"/>
6030 </complexType>
6031 </element>
6032 <element name="DocExchange">
6033 <complexType>
6034 <sequence>
6035 <element ref="tns:ebXMLSenderBinding" minOccurs="0"/>
6036 <element ref="tns:ebXMLReceiverBinding" minOccurs="0"/>
6037 </sequence>
6038 <attribute name="docExchangeId" type="ID" use="required"/>
6039 </complexType>
6040 </element>
6041 <element name="ReliableMessaging">
6042 <complexType>
6043 <sequence>
6044 <element name="Retries" type="integer" minOccurs="0"/>
6045 <element name="RetryInterval" type="duration" minOccurs="0"/>
6046 <element name="MessageOrderSemantics" type="tns:messageOrderSemantics.type"/>
6047 </sequence>
6048 </complexType>
6049 </element>
6050 <element name="PersistDuration" type="duration"/>
6051 <element name="SenderNonRepudiation">
6052 <complexType>
6053 <sequence>
6054 <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
6055 <element ref="tns:HashFunction"/>
6056 <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>

```

```

6057         <element name="SigningCertificateRef" type="tns:CertificateRef.type"/>
6058     </sequence>
6059 </complexType>
6060 </element>
6061 <element name="ReceiverNonRepudiation">
6062     <complexType>
6063         <sequence>
6064             <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
6065             <element ref="tns:HashFunction"/>
6066             <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>
6067             <element name="SigningSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6068 minOccurs="0"/>
6069         </sequence>
6070     </complexType>
6071 </element>
6072 <element name="HashFunction" type="tns:non-empty-string"/>
6073 <element name="EncryptionAlgorithm">
6074     <complexType>
6075         <simpleContent>
6076             <extension base="tns:non-empty-string">
6077                 <attribute name="minimumStrength" type="integer"/>
6078                 <attribute name="oid" type="tns:non-empty-string"/>
6079                 <attribute name="w3c" type="tns:non-empty-string"/>
6080                 <attribute name="enumerationType" type="tns:non-empty-string"/>
6081             </extension>
6082         </simpleContent>
6083     </complexType>
6084 </element>
6085 <element name="SignatureAlgorithm">
6086     <complexType>
6087         <simpleContent>
6088             <extension base="tns:non-empty-string">
6089                 <attribute name="oid" type="tns:non-empty-string"/>
6090                 <attribute name="w3c" type="tns:non-empty-string"/>
6091                 <attribute name="enumerationType" type="tns:non-empty-string"/>
6092             </extension>
6093         </simpleContent>
6094     </complexType>
6095 </element>
6096 <element name="SenderDigitalEnvelope">
6097     <complexType>
6098         <sequence>
6099             <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
6100             <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
6101             <element name="EncryptionSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
6102 minOccurs="0"/>
6103         </sequence>
6104     </complexType>
6105 </element>
6106 <element name="ReceiverDigitalEnvelope">
6107     <complexType>
6108         <sequence>
6109             <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
6110             <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
6111             <element name="EncryptionCertificateRef" type="tns:CertificateRef.type"/>
6112         </sequence>
6113     </complexType>
6114 </element>
6115 <element name="ebXMLSenderBinding">
6116     <complexType>
6117         <sequence>
6118             <element ref="tns:ReliableMessaging" minOccurs="0"/>
6119             <element ref="tns:PersistDuration" minOccurs="0"/>
6120             <element ref="tns:SenderNonRepudiation" minOccurs="0"/>
6121             <element ref="tns:SenderDigitalEnvelope" minOccurs="0"/>
6122             <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
6123         </sequence>
6124         <attribute ref="tns:version" use="required"/>
6125     </complexType>
6126 </element>
6127 <element name="ebXMLReceiverBinding">

```

```

6128     <complexType>
6129       <sequence>
6130         <element ref="tns:ReliableMessaging" minOccurs="0" />
6131         <element ref="tns:PersistDuration" minOccurs="0" />
6132         <element ref="tns:ReceiverNonRepudiation" minOccurs="0" />
6133         <element ref="tns:ReceiverDigitalEnvelope" minOccurs="0" />
6134         <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded" />
6135       </sequence>
6136       <attribute ref="tns:version" use="required" />
6137     </complexType>
6138 </element>
6139 <element name="NamespaceSupported">
6140   <complexType>
6141     <simpleContent>
6142       <extension base="anyURI">
6143         <attribute name="location" type="anyURI" use="required" />
6144         <attribute ref="tns:version" />
6145       </extension>
6146     </simpleContent>
6147   </complexType>
6148 </element>
6149 <element name="BusinessTransactionCharacteristics">
6150   <complexType>
6151     <attribute name="isNonRepudiationRequired" type="boolean" />
6152     <attribute name="isNonRepudiationReceiptRequired" type="boolean" />
6153     <attribute name="isSecureTransportRequired" type="boolean" />
6154     <attribute name="isConfidential" type="tns:persistenceLevel.type" />
6155     <attribute name="isAuthenticated" type="tns:persistenceLevel.type" />
6156     <attribute name="isTamperProof" type="tns:persistenceLevel.type" />
6157     <attribute name="isAuthorizationRequired" type="boolean" />
6158     <attribute name="isIntelligibleCheckRequired" type="boolean" />
6159     <attribute name="timeToAcknowledgeReceipt" type="duration" />
6160     <attribute name="timeToAcknowledgeAcceptance" type="duration" />
6161     <attribute name="timeToPerform" type="duration" />
6162     <attribute name="retryCount" type="integer" />
6163   </complexType>
6164 </element>
6165 <element name="MessagingCharacteristics">
6166   <complexType>
6167     <attribute ref="tns:syncReplyMode" default="none" />
6168     <attribute name="ackRequested" type="tns:perMessageCharacteristics.type"
6169 default="perMessage" />
6170     <attribute name="ackSignatureRequested" type="tns:perMessageCharacteristics.type"
6171 default="perMessage" />
6172     <attribute name="duplicateElimination" type="tns:perMessageCharacteristics.type"
6173 default="perMessage" />
6174     <attribute name="actor" type="tns:actor.type" />
6175   </complexType>
6176 </element>
6177 <element name="ServiceBinding">
6178   <complexType>
6179     <sequence>
6180       <element ref="tns:Service" />
6181       <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded" />
6182       <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded" />
6183     </sequence>
6184   </complexType>
6185 </element>
6186 <element name="CanSend">
6187   <complexType>
6188     <sequence>
6189       <element name="ThisPartyActionBinding" type="tns:ActionBinding.type" />
6190       <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0" />
6191       <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded" />
6192     </sequence>
6193   </complexType>
6194 </element>
6195 <element name="CanReceive">
6196   <complexType>
6197     <sequence>
6198       <element name="ThisPartyActionBinding" type="tns:ActionBinding.type" />

```

```
6199         <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0" />
6200         <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded" />
6201     </sequence>
6202 </complexType>
6203 </element>
6204 <element name="Status">
6205     <complexType>
6206         <attribute name="value" type="tns:statusValue.type" use="required" />
6207     </complexType>
6208 </element>
6209 <element name="Start" type="dateTime" />
6210 <element name="End" type="dateTime" />
6211 <element name="Type" type="tns:non-empty-string" />
6212 <element name="ConversationConstraints">
6213     <complexType>
6214         <attribute name="invocationLimit" type="int" />
6215         <attribute name="concurrentConversations" type="int" />
6216     </complexType>
6217 </element>
6218 <element name="Role">
6219     <complexType>
6220         <attribute name="name" type="tns:non-empty-string" use="required" />
6221         <attributeGroup ref="tns:xlink.grp" />
6222     </complexType>
6223 </element>
6224 <element name="SignatureTransforms">
6225     <complexType>
6226         <sequence>
6227             <element ref="ds:Transform" maxOccurs="unbounded" />
6228         </sequence>
6229     </complexType>
6230 </element>
6231 <element name="EncryptionTransforms">
6232     <complexType>
6233         <sequence>
6234             <element ref="ds:Transform" maxOccurs="unbounded" />
6235         </sequence>
6236     </complexType>
6237 </element>
6238 <element name="Constituent">
6239     <complexType>
6240         <sequence>
6241             <element ref="tns:SignatureTransforms" minOccurs="0" />
6242             <element ref="tns:EncryptionTransforms" minOccurs="0" />
6243         </sequence>
6244         <attribute ref="tns:idref" use="required" />
6245         <attribute name="excludedFromSignature" type="boolean" default="false" />
6246         <attribute name="minOccurs" type="nonNegativeInteger" />
6247         <attribute name="maxOccurs" type="nonNegativeInteger" />
6248     </complexType>
6249 </element>
6250 <element name="Packaging">
6251     <complexType>
6252         <sequence>
6253             <element name="ProcessingCapabilities">
6254                 <complexType>
6255                     <attribute name="parse" type="boolean" use="required" />
6256                     <attribute name="generate" type="boolean" use="required" />
6257                 </complexType>
6258             </element>
6259             <element name="CompositeList" maxOccurs="unbounded">
6260                 <complexType>
6261                     <choice maxOccurs="unbounded">
6262                         <element name="Encapsulation">
6263                             <complexType>
6264                                 <sequence>
6265                                     <element ref="tns:Constituent" />
6266                                 </sequence>
6267                                 <attributeGroup ref="tns:pkg.grp" />
6268                             </complexType>
6269                         </element>
```

```

6270         <element name="Composite">
6271             <complexType>
6272                 <sequence>
6273                     <element ref="tns:Constituent" maxOccurs="unbounded"/>
6274                 </sequence>
6275                 <attributeGroup ref="tns:pkg.grp"/>
6276             </complexType>
6277         </element>
6278     </choice>
6279 </complexType>
6280 </element>
6281 </sequence>
6282 <attribute ref="tns:id" use="required"/>
6283 </complexType>
6284 </element>
6285 <element name="Comment">
6286     <complexType>
6287         <simpleContent>
6288             <extension base="tns:non-empty-string">
6289                 <attribute ref="xml:lang"/>
6290             </extension>
6291         </simpleContent>
6292     </complexType>
6293 </element>
6294 <element name="SimplePart">
6295     <complexType>
6296         <sequence>
6297             <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
6298         </sequence>
6299         <attributeGroup ref="tns:pkg.grp"/>
6300         <attribute ref="xlink:role"/>
6301     </complexType>
6302 </element>
6303 <!-- COMMON -->
6304 <simpleType name="statusValue.type">
6305     <restriction base="NMTOKEN">
6306         <enumeration value="agreed"/>
6307         <enumeration value="signed"/>
6308         <enumeration value="proposed"/>
6309     </restriction>
6310 </simpleType>
6311 <simpleType name="endpointType.type">
6312     <restriction base="NMTOKEN">
6313         <enumeration value="login"/>
6314         <enumeration value="request"/>
6315         <enumeration value="response"/>
6316         <enumeration value="error"/>
6317         <enumeration value="allPurpose"/>
6318     </restriction>
6319 </simpleType>
6320 <simpleType name="non-empty-string">
6321     <restriction base="string">
6322         <minLength value="1"/>
6323     </restriction>
6324 </simpleType>
6325 <simpleType name="syncReplyMode.type">
6326     <restriction base="NMTOKEN">
6327         <enumeration value="mshSignalsOnly"/>
6328         <enumeration value="responseOnly"/>
6329         <enumeration value="signalsAndResponse"/>
6330         <enumeration value="signalsOnly"/>
6331         <enumeration value="none"/>
6332     </restriction>
6333 </simpleType>
6334 <complexType name="service.type">
6335     <simpleContent>
6336         <extension base="tns:non-empty-string">
6337             <attribute name="type" type="tns:non-empty-string"/>
6338         </extension>
6339     </simpleContent>
6340 </complexType>

```

```

6341 <complexType name="protocol.type">
6342   <simpleContent>
6343     <extension base="tns:non-empty-string">
6344       <attribute ref="tns:version"/>
6345     </extension>
6346   </simpleContent>
6347 </complexType>
6348 <attribute name="idref" type="IDREF"/>
6349 <attribute name="id" type="ID"/>
6350 <attribute name="version" type="tns:non-empty-string"/>
6351 <attribute name="syncReplyMode" type="tns:syncReplyMode.type"/>
6352 <complexType name="SecurityPolicy.type"/>
6353 <complexType name="CertificateRef.type">
6354   <attribute name="certId" type="IDREF" use="required"/>
6355 </complexType>
6356 <simpleType name="perMessageCharacteristics.type">
6357   <restriction base="NMTOKEN">
6358     <enumeration value="always"/>
6359     <enumeration value="never"/>
6360     <enumeration value="perMessage"/>
6361   </restriction>
6362 </simpleType>
6363 <simpleType name="actor.type">
6364   <restriction base="NMTOKEN">
6365     <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:nextMSH"/>
6366     <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"/>
6367   </restriction>
6368 </simpleType>
6369 <simpleType name="messageOrderSemantics.type">
6370   <restriction base="Name">
6371     <enumeration value="Guaranteed"/>
6372     <enumeration value="NotGuaranteed"/>
6373   </restriction>
6374 </simpleType>
6375 <complexType name="SecurityDetailsRef.type">
6376   <attribute name="securityId" type="IDREF" use="required"/>
6377 </complexType>
6378 <simpleType name="persistenceLevel.type">
6379   <restriction base="Name">
6380     <enumeration value="none"/>
6381     <enumeration value="transient"/>
6382     <enumeration value="persistent"/>
6383     <enumeration value="transient-and-persistent"/>
6384   </restriction>
6385 </simpleType>
6386 <element name="SecurityDetailsRef" type="tns:SecurityDetailsRef.type"/>
6387 <element name="SecurityDetails">
6388   <complexType>
6389     <sequence>
6390       <element ref="tns:TrustAnchors" minOccurs="0"/>
6391       <element ref="tns:SecurityPolicy" minOccurs="0"/>
6392     </sequence>
6393     <attribute name="securityId" type="ID" use="required"/>
6394   </complexType>
6395 </element>
6396 <element name="TrustAnchors">
6397   <complexType>
6398     <sequence>
6399       <element name="AnchorCertificateRef" type="tns:CertificateRef.type"
6400 maxOccurs="unbounded"/>
6401     </sequence>
6402   </complexType>
6403 </element>
6404 <element name="SecurityPolicy">
6405   <complexType>
6406     <sequence>
6407     </sequence>
6408   </complexType>
6409 </element>
6410 <simpleType name="accessAuthentication.type">
6411   <restriction base="NMTOKEN">

```



```
6412         <enumeration value="basic"/>
6413         <enumeration value="digest"/>
6414     </restriction>
6415 </simpleType>
6416 </schema>
6417
```

## 6418 Appendix E CPA Composition (Non-Normative)

### 6419 E.1 Suggestions for Design of Computational Procedures

6420 A quick inspection of the schemas for the top level elements, *CollaborationProtocolProfile*  
6421 (*CPP*) and *CollaborationProtocolAgreement* (*CPA*), shows that a *CPA* can be viewed as a  
6422 result of merging portions of the *PartyInfo* elements found in constituent *CPPs*, and then  
6423 integrating these *PartyInfo* elements with other *CPA* sibling elements, such as those governing  
6424 the *CPA* validity period.

6425  
6426 Merging *CPPs* into *CPAs* is one way in which trading partners can arrive at a proposed or  
6427 “draft” *CPA*. A draft *CPA* might also be formed from a *CPA* template. A *CPA*-template  
6428 represents one party’s proposed implementation of a business process that uses placeholder  
6429 values for the identifying aspects of the other party, such as *PartyId* or *TransportEndpoint*  
6430 elements. To form a *CPA* from a *CPA* template, the placeholder values are replaced by the actual  
6431 values for the other trading partner. The actual values could themselves be extracted from the  
6432 other trading partner’s *CPP*, if one is available, or they could be obtained from an administrator  
6433 performing data entry functions.

6434  
6435 We call objects draft *CPAs* to indicate their potential use as inputs to a *CPA* negotiation process  
6436 in which a draft *CPA* is verified as suitable for both parties, modified until a suitable *CPA* is  
6437 found, or discovered to not be feasible until one side (or both) acquires additional software  
6438 capabilities. These negotiation procedures and protocols are currently being designed, their  
6439 requirements having been defined, and the resulting specifications should be available with the  
6440 next release of this specification. In general, a draft *CPA* will constitute a proposal about an  
6441 overall binding of a business process to a delivery implementation, while negotiation will be  
6442 used to arrive at detailed values for parameters reflecting a final agreement. A special companion  
6443 document, the *NegotiationDescriptorDocument*, provides both focus on what parameters can be  
6444 negotiated as well as ranges or sets of acceptable values for those parameters.

6445  
6446 In the remainder of this appendix, the goal will be to identify and describe the basic tasks that  
6447 computational procedures for the assembly of the draft *CPA* would normally accomplish. While  
6448 no normative specification is provided for an algorithm for *CPA* formation, some guidance for  
6449 implementers is provided. This information might assist the software implementer in designing a  
6450 partially automated and partially interactive software system useful for configuring business  
6451 collaboration so as to arrive at satisfactorily complete levels of interoperability.

6452  
6453 Before enumerating and describing the basic tasks, it is worthwhile mentioning  
6454 two basic reasons why we focus on the component tasks involved in *CPA* formation rather than  
6455 attempt to provide an algorithm for *CPA* formation. These reasons provide some hints to  
6456 implementers about ways in which they might customize their approaches to drafting *CPAs* from  
6457 *CPPs*.

#### 6458 E.1.1 Variability in Inputs

6460 User preferences provide one source of variability in the inputs to the *CPA* formation process.  
6461 Let us suppose in this section that each of the *Parties* has made its *CPP* available to potential

6462 collaborators. Normally one *Party* will have a desired business collaboration (defined in a  
6463 ***ProcessSpecification*** document) to implement with its intended collaborator. So the information  
6464 inputs will normally involve a user preference about intended business collaboration in addition  
6465 to just the *CPPs*.

6466  
6467 A *CPA* formation tool can have access to local user information not advertised in the *CPP* that  
6468 can contribute to the *CPA* that is formed. A user can have chosen to only advertise those system  
6469 capabilities that reflect capabilities that have not been deprecated. For example, a user can only  
6470 advertise HTTP and omit FTP, even when capable of using FTP. The reason for omitting FTP  
6471 might be concerns about the scalability of managing user accounts, directories, and passwords  
6472 for FTP sessions. Despite not advertising an FTP capability, configuration software can use tacit  
6473 knowledge about its own FTP capability to form a *CPA* with an intended collaborator who  
6474 happens to have only an FTP capability for implementing a desired business collaboration. In  
6475 other words, business interests can, in this case, override the deprecation policy. Both tacit  
6476 knowledge and detailed preference information account for variability in inputs into the *CPA*  
6477 formation process.

### 6478 6479 **E.1.2 Variable Stringency in Evaluating Proposed Agreements**

6480 The conditions for output of a *CPA* given two *CPPs* can involve different levels and extents of  
6481 interoperability. In other words, when an optimal solution that satisfies every level of  
6482 requirement and every other additional constraint does not exist, a *Party* can propose a *CPA* that  
6483 satisfies enough of the requirements for "a good enough" implementation. User input can be  
6484 solicited to determine what is a good enough implementation, and so can be as varied as there  
6485 are user configuration options to express preferences. In practice, compromises can be made on  
6486 security, reliable messaging, levels of signals and acknowledgments, and other matters in order  
6487 to find some acceptable means of doing business.

6488  
6489 A *CPA* can support a fully interoperable configuration in which agreement has been reached on  
6490 all technical levels needed for business collaboration. In such a case, matches in capabilities will  
6491 have been found in all relevant technical levels.

6492  
6493 However, there can be interoperable configurations agreed to in a *CPA* in which not all aspects  
6494 of a business collaboration match. Gaps can exist in packaging, security, signaling, reliable  
6495 messaging and other areas and yet the systems can still transport the business data, and special  
6496 means can be employed to handle the exceptions. In such situations, a *CPA* can reflect  
6497 configured policies or expressly solicited user permission to ignore some shortcomings in  
6498 configurations. A system might not be capable of responding in a business collaboration so as to  
6499 support a specified ability to supply non-repudiation of receipt, but might still be acceptable for  
6500 business reasons. A system might not be able to handle all the processing needed to support, for  
6501 example, SOAP with Attachments and yet still be able to treat the multipart according to  
6502 "multipart/mixed" handling and allow business collaboration to take place. In fact, short of a  
6503 failure to be able to transport data and a failure to be able to provide data relevant to the business  
6504 collaboration, there are few features that might not be temporarily or indefinitely compromised  
6505 about, given overriding *business* interests. This situation of "partial interoperability" is to be  
6506 expected to persist for some time, and so interferes with formulating a "clean" algorithm for  
6507 deciding on what is sufficient for interoperability.

6508

## 6509 E.2 CPA Formation Component Tasks

6510 Technically viewed, a *CPA* provides "bindings" between business collaboration specifications  
6511 (such as those defined within the *ProcessSpecification*'s referenced documents) and those  
6512 services and protocols that are used to implement these specifications. The implementation takes  
6513 place at several levels and involves varied services at these levels. A *CPA* that arrives at a fully  
6514 interoperable collaboration binding can be thought of as arriving at interoperable, application-to-  
6515 application integration. *CPAs* can fall short of this goal and still be both useful and acceptable to  
6516 the collaborating *parties*. Certainly, if no matching data-transport capabilities can be discovered,  
6517 a *CPA* would not provide much in the way of interoperable integration. Likewise, partial *CPAs*  
6518 can leave significant system work to be done before a completely satisfactory application-to-  
6519 application integration is realized. Even so, partial integration can be sufficient to allow  
6520 collaboration, and to enjoy payoffs from increased levels of automation.

6521  
6522 In practice, the *CPA* formation process can produce a complete *CPA*, a failure result, a gap list  
6523 that drives a dialog with the user, or perhaps even a *CPA* that implements partial interoperability  
6524 "good enough" for the business collaborators. Because both matching capabilities and  
6525 interoperability can be matters of degree, the constituent tasks are finding the matches in  
6526 capabilities at different levels and for different services. We next proceed to characterize the  
6527 most important of these constituent tasks.  
6528

## 6529 E.3 CPA Formation from *CPPs*: Context of Tasks

6530 To simplify discussion, assume in the following that we are viewing the tasks faced by a  
6531 software agent when:  
6532

- 6533 1. an intended collaborator is known and the collaborator's *CPP* has been retrieved,
- 6534 2. the *ProcessSpecification* between our side and our intended collaborator has been  
6535 selected,
- 6536 3. the *Service*, *Action*, and the specific *Role* elements that our software agent is to play in  
6537 the business collaboration is known, and
- 6538 4. finally, the capabilities that we have advertised in our *CPP* are known  
6539

6540 For vividness, we will develop our discussions using the "3A4" ebBPSS example and the *CPPs*  
6541 of Company A and B that are found in full in appendices of this document and that should also  
6542 be available at the web site for the OASIS ebXML CPPA Technical Committee. For simplicity,  
6543 we will assume that the information about capabilities is restricted to what is available in our  
6544 agent's *CPP*, and in the *CPP* of our intended collaborator. We will suppose that we have taken  
6545 on the viewpoint of Company A assembling a draft *CPA*. Please note that there is no guarantee  
6546 that the same draft *CPAs* will be produced in the same order from differing viewpoints.  
6547

6548 In general, the basic tasks consist of finding "matches" between our capabilities and our intended  
6549 collaborator's capabilities at the various levels of the collaboration protocol stack and with  
6550 respect to the services supplied at these various levels. This stack, which need not be  
6551 characterized in any detail, is at least distinguished by an application level and a messaging  
6552 transfer level. The application level is governed by a business process flow specification, such as  
6553 ebBPSS. The messaging transfer level will consist of a number of requirements and options

6554 concerning transfer protocols, security, packaging, and messaging patterns (such as various kinds  
6555 of acknowledgment, error messages, and the like.)

6556  
6557 In actually assembling the tasks into a computational process, it will generally make sense to  
6558 perform the tasks in a certain order. The overall order reflects the implicit structure of the *CPA*:  
6559 first undertake those tasks to ensure that there is a match with respect to the business  
6560 collaboration process. Without finding that the collaborators can participate in the same  
6561 **ProcessSpecification** successfully, there is little point in working through implementation  
6562 options. Then, examine the matches within the components of the bindings that have been  
6563 announced for the business collaboration process, checking for the most indispensable “matches”  
6564 first (**Transport**-related), and continuing checks on the other layers reflecting integrated  
6565 interoperability at packaging, security, signals and protocol patterns, and so on. With this basic  
6566 overview in mind, let us proceed to consider the basic tasks in greater detail.

6567

#### 6568 **E.4 Business Collaboration Process Matching Tasks**

6569 Company A has announced within its *CPP*, at least one **PartyInfo** element. For current purposes,  
6570 the most important initial focus is on all the sibling elements with the path  
6571 **/CollaborationProtocolProfile/PartyInfo/CollaborationRole**. Each element of this kind has a  
6572 child, **ProcessSpecification**. Our initial matching task (probably better viewed as a filtering  
6573 task) is to select those nodes where the **ProcessSpecification** is one that we are interested in  
6574 building a CPA for! Checking the attribute values allows us to select by comparing values in the  
6575 **name**, **xlink:href** or **uuid** attributes. The definitive value for matching **ebBPSS** process  
6576 specifications is the value found in the **ProcessSpecification/@uuid** attribute.

6577

##### 6578 **E.4.1 Matching ProcessSpecification/Roles, and Actions: Initial Filtering and Selection**

6579 The previous task has essentially found two **CollaborationRole** node sets within our and our  
6580 collaborator’s *CPP* documents where the **ProcessSpecifications** are identical, and equal to the  
6581 value of interest given above. In other words, we have **CollaborationRoles** with  
6582 **ProcessSpecification/@name=’PIP3A4RequestPurchaseOrder’**. It is convenient but not essential  
6583 to use the **name** attribute in performing this selection.

6584

6585 We next proceed to filter these node sets. We have been given our **Role** element value for our  
6586 participation in the **ProcessSpecification**. For Company A, this **Role** has the **name** attribute with  
6587 value ‘Buyer’. Because we are here considering only **BinaryCollaborations** in **ebBPSS**  
6588 terminology (or their equivalent in other flow languages), we are only interested in those  
6589 **CollaborationRole** node sets within our collaborator’s *CPP* that have a **Role** value equal to  
6590 ‘Seller.’ So we assume we have narrowed our focus to **CollaborationRole** node sets in Company  
6591 A’s *CPP* with **Role/@name=’Buyer’** and in Company B’s **CollaborationRole** node sets with  
6592 **Role/@name=’Seller’**.

6593

6594 For more general collaborations, such as in the **MultiPartyCollaborations** of **ebBPSS**, we would  
6595 need to know the list of roles available within the process, and keep track of that for each of the  
6596 **CollaborationRoles**, the **Role** values chosen correspond correctly for the participants. We do not  
6597 here discuss the matching/filtering task for collaborations involving more than two roles, as  
6598 multiparty *CPAs* are not within scope for version 2.0 of this specification.

6599

## 6600 E.5 Implementation Matching Tasks

6601 After filtering the CollaborationRoles with the desired ProcessSpecification, we should find one  
6602 CollaborationRole in our own CPP where we play the Buyer role, and one CollaborationRole in  
6603 our intended collaborator Company B's CPP where it plays the Seller role.

6604  
6605 Our next task is to locate the specific candidate *bindings* relevant to CPA formation. There are  
6606 bindings for Service and Actions. For initial simplicity, we consider detailed matching tasks as  
6607 they arise for a standard collaboration case involving a *Request* action, followed by a *Response*  
6608 action. For version 2.0 of this specification, most matching tasks will involve matching of  
6609 referenced components of the CPP's *ThisPartyActionBinding* elements under  
6610 *CollaborationRole/ServiceBinding/CanSend/* and under  
6611 *CollaborationRole/ServiceBinding/CanReceive*.

### 6613 E.5.1 Action Correspondence and Selecting Correlative PackageIds, and ChannelIds

6614 In CPPs, under each of the elements, *CollaborationRole/ServiceBinding/CanSend* and  
6615 *CollaborationRole/ServiceBinding/CanReceive*, are lists of *ThisPartyActionBindings*. For  
6616 *Request-Response* collaboration patterns, we are interested in matches:

- 6617 1. in the bindings of the Requesting side's *CanSend/ThisPartyActionBinding* with the  
6618 Responding side's *CanReceive/ThisPartyActionBinding* for the request action, and
- 6619 2. in the bindings of the Responding side's *CanSend/ThisPartyActionBinding* with the  
6620 Requesting side's *CanReceive/ThisPartyActionBinding* for the response action

6621  
6622  
6623 These correlative bindings give us references to detailed components that need to match for a  
6624 fully interoperable agreement. Case 1 pertains to the *Request*. Case 2 pertains to the *Response*.

6625  
6626 For example, for Company A, we find under *CanSend*:

```
6627 <tp:ThisPartyActionBinding tp:action="Purchase Order Request Action"
6628 tp:packageId="CompanyA_RequestPackage">
6629   <tp:BusinessTransactionCharacteristics ... />
6630   <tp:ActionContext tp:binaryCollaboration="Request Purchase Order"
6631   tp:businessTransactionActivity="Request Purchase Order"
6632   tp:requestOrResponseAction="Purchase Order Request Action"/>
6633   <tp:ChannelId>asyncChannelA1</tp:ChannelId>
6634 </tp:ThisPartyActionBinding>
```

6635  
6636  
6637 Correlative to this, for Company B, we find under *CanReceive*:

```
6638 <tp:ThisPartyActionBinding tp:action="Purchase Order Request Action"
6639 tp:packageId="CompanyB_RequestPackage">
6640   <tp:BusinessTransactionCharacteristics ... />
6641   <tp:ActionContext tp:binaryCollaboration="Request Purchase Order"
6642   tp:businessTransactionActivity="Request Purchase Order"
6643   tp:requestOrResponseAction="Purchase Order Request Action"/>
6644   <tp:ChannelId>asyncChannelB1</tp:ChannelId>
6645 </tp:ThisPartyActionBinding>
```

6646  
6647  
6648 The correlation of elements can normally (when we are dealing with BPSS *Binary*

6649 *Collaborations* or their equivalents in other representations) be based on equality of the *action*  
 6650 (or *requestOrResponseAction*) values. More detailed correlation of elements can make use of  
 6651 more detailed testing and comparisons of the values in the *ActionContext* child elements of the  
 6652 relevant *CanSend* and *CanReceive* pairs.

6653  
 6654 In the preceding, we have illustrated the matching of *CanSend* and *CanReceive* for  
 6655 asynchronous bindings. All *CanSend* bindings that are siblings under a *ServiceBinding* element  
 6656 are asynchronous and make use of separate TCP connections that the *CanSend* side initiates on a  
 6657 listening TCP port. In order to represent binding details for synchronous sending, the convention  
 6658 is adopted whereby the *CanSend* element for a Receiver is placed under its *CanReceive* element.  
 6659 This is illustrated by:

```

6660 <tp:CanSend>
6661   <tp:ThisPartyActionBinding
6662     tp:id="companyA_ABID6"
6663     tp:action="Purchase Order Request Action"
6664     tp:packageId="CompanyA_RequestPackage">
6665   <tp:BusinessTransactionCharacteristics
6666     tp:isNonRepudiationRequired="true"
6667     tp:isNonRepudiationReceiptRequired="true"
6668     tp:isSecureTransportRequired="true"
6669     tp:isConfidential="transient"
6670     tp:isAuthenticated="persistent"
6671     tp:isTamperProof="persistent"
6672     tp:isAuthorizationRequired="true"
6673     tp:timeToAcknowledgeReceipt="PT2H"
6674     tp:timeToPerform="P1D" />
6675   <tp:ActionContext
6676     tp:binaryCollaboration="Request Purchase Order"
6677     tp:businessTransactionActivity="Request Purchase Order"
6678     tp:requestOrResponseAction="Purchase Order Request Action"/>
6679   <tp:ChannelId>syncChannelA1</tp:ChannelId>
6680 </tp:ThisPartyActionBinding>
6681 <tp:CanReceive>
6682   <tp:ThisPartyActionBinding
6683     tp:id="companyA_ABID7"
6684     tp:action="Purchase Order Confirmation Action"
6685     tp:packageId="CompanyA_SyncReplyPackage">
6686   <tp:BusinessTransactionCharacteristics
6687     tp:isNonRepudiationRequired="true"
6688     tp:isNonRepudiationReceiptRequired="true"
6689     tp:isSecureTransportRequired="true"
6690     tp:isConfidential="transient"
6691     tp:isAuthenticated="persistent"
6692     tp:isTamperProof="persistent"
6693     tp:isAuthorizationRequired="true"
6694     tp:timeToAcknowledgeReceipt="PT2H"
6695     tp:timeToPerform="P1D" />
6696   <tp:ActionContext
6697     tp:binaryCollaboration="Request Purchase Order"
6698     tp:businessTransactionActivity="Request Purchase Order"
6699     tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
6700   <tp:ChannelId>syncChannelA1</tp:ChannelId>
6701 </tp:ThisPartyActionBinding>
6702

```

```

6703     </tp:CanReceive>
6704     <tp:CanReceive>
6705         <tp:ThisPartyActionBinding
6706             tp:id="companyA_ABID8"
6707             tp:action="Exception"
6708             tp:packageId="CompanyA_ExceptionPackage">
6709             <tp:BusinessTransactionCharacteristics
6710                 tp:isNonRepudiationRequired="true"
6711                 tp:isNonRepudiationReceiptRequired="true"
6712                 tp:isSecureTransportRequired="true"
6713                 tp:isConfidential="transient"
6714                 tp:isAuthenticated="persistent"
6715                 tp:isTamperProof="persistent"
6716                 tp:isAuthorizationRequired="true"
6717                 tp:timeToAcknowledgeReceipt="PT2H"
6718                 tp:timeToPerform="P1D"/>
6719             <tp:ChannelId>syncChannelA1</tp:ChannelId>
6720         </tp:ThisPartyActionBinding>
6721     </tp:CanReceive>
6722 </tp:CanSend>
6723

```

6724 This subordination will also carry over to the synchronous receiving side, in which its  
6725 **CanReceive** element(s) is (are) under the **CanSend** element used to represent the initial sending  
6726 of a request. An illustration from Company B's synchronous binding is:

```

6727 <tp:CanReceive>
6728     <tp:ThisPartyActionBinding
6729         tp:id="companyB_ABID8"
6730         tp:action="Purchase Order Request Action"
6731         tp:packageId="CompanyB_SyncReplyPackage">
6732     <tp:BusinessTransactionCharacteristics
6733         tp:isNonRepudiationRequired="true"
6734         tp:isNonRepudiationReceiptRequired="true"
6735         tp:isSecureTransportRequired="true" tp:isConfidential="transient"
6736         tp:isAuthenticated="persistent" tp:isTamperProof="persistent"
6737         tp:isAuthorizationRequired="true" tp:timeToAcknowledgeReceipt="PT5M"
6738         tp:timeToPerform="PT5M"/>
6739     <tp:ActionContext
6740         tp:binaryCollaboration="Request Purchase Order"
6741         tp:businessTransactionActivity="Request Purchase Order"
6742         tp:requestOrResponseAction="Purchase Order Request Action"/>
6743     <tp:ChannelId>syncChannelB1</tp:ChannelId>
6744 </tp:ThisPartyActionBinding>
6745 <tp:CanSend>
6746     <tp:ThisPartyActionBinding
6747         tp:id="companyB_ABID6"
6748         tp:action="Purchase Order Confirmation Action"
6749         tp:packageId="CompanyB_ResponsePackage">
6750     <tp:BusinessTransactionCharacteristics
6751         tp:isNonRepudiationRequired="true"
6752         tp:isNonRepudiationReceiptRequired="true"
6753         tp:isSecureTransportRequired="true"
6754         tp:isConfidential="transient"
6755         tp:isAuthenticated="persistent"
6756         tp:isTamperProof="persistent"
6757

```



```

6758         tp:isAuthorizationRequired="true"
6759         tp:timeToAcknowledgeReceipt="PT5M"
6760         tp:timeToPerform="PT5M" />
6761     <tp:ActionContext
6762         tp:binaryCollaboration="Request Purchase Order"
6763         tp:businessTransactionActivity="Request Purchase Order"
6764         tp:requestOrResponseAction="Purchase Order Confirmation Action" />
6765     <tp:ChannelId>syncChannelB1</tp:ChannelId>
6766 </tp:ThisPartyActionBinding>
6767 </tp:CanSend>
6768 <tp:CanSend>
6769     <tp:ThisPartyActionBinding
6770         tp:id="companyB_ABID7"
6771         tp:action="Exception"
6772         tp:packageId="CompanyB_ExceptionPackage">
6773 <tp:BusinessTransactionCharacteristics
6774     tp:isNonRepudiationRequired="true"
6775     tp:isNonRepudiationReceiptRequired="true"
6776     tp:isSecureTransportRequired="true"
6777     tp:isConfidential="transient"
6778     tp:isAuthenticated="persistent"
6779     tp:isTamperProof="persistent"
6780     tp:isAuthorizationRequired="true"
6781     tp:timeToAcknowledgeReceipt="PT5M"
6782     tp:timeToPerform="PT5M" />
6783 <tp:ChannelId>syncChannelB1</tp:ChannelId>
6784 </tp:ThisPartyActionBinding>
6785 </tp:CanSend>
6786 </tp:CanReceive>
6787

```

### 6788 **E.5.2 Matching and Checking DeliveryChannel Details**

6789 Until now, most of the matching work has been undertaken to find pairs of correlative action  
6790 binding, and so the matching has functioned as a filtering mechanism. Once in possession of  
6791 pairs of correlative action bindings, however, the work of checking for matches across the  
6792 various dimensions of operation—transport, transport security, PKI compatibility for various  
6793 tasks, agreement about messaging characteristics (reliable messaging, digital enveloping, signed  
6794 acknowledgments (minimal non-repudiation of receipt), non-repudiation of origin, packaging  
6795 details, and more begins.

6796  
6797 Once in possession of the action bindings, IDREFs provide references to the underlying  
6798 components for comparison. For example, when comparing packaging details, the *Request*  
6799 IDREFS are found at *CanSend/ThisPartyActionBinding/@packageId* and within the other *CPP*  
6800 at *CanReceive/ThisPartyActionBinding@packageId*. For Company A's *Request* "Purchase  
6801 Order Request Action," the packaging IDREF is found in:

```
6803 tp:packageId="CompanyA_RequestPackage"
```

6805 and this IDREF value refers to:

```

6807 <tp:Packaging tp:id="CompanyA_RequestPackage">
6808     <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
6809     <tp:CompositeList>
6810         <tp:Composite tp:id="CompanyA_RequestMsg"

```

```

6811         tp:mimetype="multipart/related" tp:mimeparameters="type=text/xml;">
6812         <tp:Constituent tp:idref="CompanyA_MsgHdr" />
6813         <tp:Constituent tp:idref="CompanyA_Request" />
6814         </tp:Composite>
6815         </tp:CompositeList>
6816 </tp:Packaging>
6817

```

6818 For Company A's *Request* "Purchase Order Request Action", the delivery channel IDREF is  
6819 found in:

```

6820 <tp:ChannelId>asyncChannelA1</tp:ChannelId>
6821
6822

```

6823 and this IDREF value refers to the element with this ID, namely:

```

6824 <tp:DeliveryChannel tp:channelId="asyncChannelA1"
6825 tp:transportId="transportA1" tp:docExchangeId="docExchangeA1">
6826 <tp:MessagingCharacteristics
6827     tp:syncReplyMode="none"
6828     tp:ackRequested="always"
6829     tp:ackSignatureRequested="always"
6830     tp:duplicateElimination="always" />
6831 </tp:DeliveryChannel>
6832
6833

```

6834 Two remaining crucial references for understanding the binding, are found in attributes of the  
6835 *DeliveryChannel*, namely: *DeliveryChannel/@transportId* and in the attribute  
6836 *DeliveryChannel/@docExchangeId*.

6837  
6838 For Company A, for example, we find *transportId*="transportA1" and  
6839 *docExchangeId*="docExchangeA1" are the IDREFs for the continuing binding information with  
6840 the *DeliveryChannel*, "asyncChannelA1". Resolving these references, we obtain:

```

6841 <tp:Transport tp:transportId="transportA1">
6842     <tp:TransportSender>
6843         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
6844         <tp:TransportClientSecurity>
6845             <tp:TransportSecurityProtocol
6846                 tp:version="3.0">SSL</tp:TransportSecurityProtocol>
6847             <ClientCertificateRef tp:certId="CompanyA_ClientCert" />
6848             <tp:ServerSecurityDetailsRef
6849                 tp:securityId="CompanyA_TransportSecurity" />
6850         </tp:TransportClientSecurity>
6851     </tp:TransportSender>
6852     <tp:TransportReceiver>
6853         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
6854         <tp:Endpoint
6855             tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/async"
6856             tp:type="allPurpose" />
6857         <tp:TransportServerSecurity>
6858             <tp:TransportSecurityProtocol
6859                 tp:version="3.0">SSL</tp:TransportSecurityProtocol>
6860             <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert" />
6861             <tp:ClientSecurityDetailsRef
6862                 tp:securityId="CompanyA_TransportSecurity" />
6863         </tp:TransportServerSecurity>
6864

```

```

6865         </tp:TransportReceiver>
6866     </tp:Transport>
6867
6868     for transportID "transportA1" and
6869
6870     <tp:DocExchange tp:docExchangeId="docExchangeA1">
6871         <tp:ebXMLSenderBinding tp:version="2.0">
6872             <tp:ReliableMessaging>
6873                 <tp:Retries>3</tp:Retries>
6874                 <tp:RetryInterval>PT2H</tp:RetryInterval>
6875                 <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
6876             </tp:ReliableMessaging>
6877             <tp:PersistDuration>P1D</tp:PersistDuration>
6878             <tp:SenderNonRepudiation>
6879                 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#
6880                 </tp:NonRepudiationProtocol>
6881                 <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1
6882                 </tp:HashFunction>
6883                 <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-sha1
6884                 </tp:SignatureAlgorithm>
6885                 <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
6886             </tp:SenderNonRepudiation>
6887             <tp:SenderDigitalEnvelope>
6888                 <tp:DigitalEnvelopeProtocol
6889                 tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
6890                 <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
6891                 <tp:EncryptionSecurityDetailsRef
6892                 tp:securityId="CompanyA_MessageSecurity"/>
6893             </tp:SenderDigitalEnvelope>
6894         </tp:ebXMLSenderBinding>
6895         <tp:ebXMLReceiverBinding tp:version="2.0">
6896             <tp:ReliableMessaging>
6897                 <tp:Retries>3</tp:Retries>
6898                 <tp:RetryInterval>PT2H</tp:RetryInterval>
6899                 <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
6900             </tp:ReliableMessaging>
6901             <tp:PersistDuration>P1D</tp:PersistDuration>
6902             <tp:ReceiverNonRepudiation>
6903                 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#
6904                 </tp:NonRepudiationProtocol>
6905                 <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1
6906                 </tp:HashFunction>
6907                 <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-sha1
6908                 </tp:SignatureAlgorithm>
6909                 <tp:SigningSecurityDetailsRef
6910                 tp:securityId="CompanyA_MessageSecurity"/>
6911             </tp:ReceiverNonRepudiation>
6912             <tp:ReceiverDigitalEnvelope>
6913                 <tp:DigitalEnvelopeProtocol
6914                 tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
6915                 <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
6916                 <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
6917             </tp:ReceiverDigitalEnvelope>
6918         </tp:ebXMLReceiverBinding>
6919     </tp:DocExchange>
6920

```

6921 for the *docExchangeId*, docExchangeA1.

6922  
6923 There are, of course, other references, such as those to security-related capabilities, that will be  
6924 important to resolve when checking detailed matching properties, but the four IDREFs (two for  
6925 the sender and two for the receiver) that have just been introduced are critical to the remainder of  
6926 the match tests that will lead to the formation of draft *CPAs*. We will assume at this point that the  
6927 reader can resolve IDREFs using the example *CPPs* and *CPAs* for Company A and B in the  
6928 appendices, and will not exhibit them in the text in order to save space.

6929  
6930 We next turn to a more in-depth treatment of the tests that are involved in finding the elements  
6931 for a draft *CPA*.

6932  
6933 The detailed tasks to be discussed in greater depth are:

- 6934  
6935 1. Matching *Channel MessagingCharacteristics*  
6936 2. Checking *BusinessTransactionCharacteristics* coherence with *Channel* details  
6937 3. Matching *Packaging*  
6938 4. Matching *Transport* and *Transport[Receiver/Sender]Security*  
6939 5. Matching and Checking *DocExchange* subtrees.  
6940

6941 Because agreement about *Transport* is quite fundamental, we shall consider it first.  
6942 Computational processes are likely to first find pairs that match on *Transport* details, and will  
6943 ignore pairs failing to have matches at this level.  
6944

#### 6945 **E.5.2.1 Matching Transport**

6946 Matching *Transport* first involves matching the *Transport/TransportSender/TransportProtocol*  
6947 capabilities of the requester with the *Transport/TransportReceiver/TransportProtocol*  
6948 capabilities found under the collaborator receiving the *request*. Several such matches can exist,  
6949 and any of these matches can be used in forming a draft, provided other aspects match up  
6950 satisfactorily. Each *CPP* is assumed to have listed its preferred transport protocols first (as  
6951 determined by the listing of the Bindings that reference the *Transport* element, but different  
6952 outcomes can result depending on which *CPP* is used first for searching for matches. In general,  
6953 resolution of preference differences is left to a distinct phase of *CPA negotiation*, following  
6954 proposal of a draft *CPA*. Negotiation can be performed by explicit actions of users, but is  
6955 expected to become increasingly automated.  
6956

6957 Matching transport secondly involves matching the *TransportSender/TransportProtocol*  
6958 capabilities of the responding collaborator with its *TransportReceiver/TransportProtocol*  
6959 capabilities found under the collaborator receiving the response, which is typically the  
6960 collaborator that has sent a request. Several such matches can exist, and any of these matches can  
6961 be used in forming a draft. In one case, however, there may be no need for the second match on  
6962 *TransportProtocol*. If we are using HTTP or some other protocol supporting synchronous  
6963 replies, and the *DeliveryChannel* has a *MessagingCharacteristics* child that has its  
6964 *syncReplyMode* attribute with a value of “signalsAndResponse,” then everything comes back  
6965 synchronously, and there is no need to match on *TransportProtocol* for the *Response*  
6966 *DeliveryChannel*.

6967  
6968 If *TransportSecurity* is present, then there can be additional checks. First,  
6969 *TransportSender/TransportClientSecurity/TransportSecurityProtocol* should be compatible  
6970 with *TransportReceiver/TransportServerSecurity/TransportSecurityProtocol*. Second, if either  
6971 the *TransportSender/TransportClientSecurity/ClientSecurityDetailsRef* or  
6972 *TransportSender/TransportClientSecurity/ServerSecurityDetailsRef* elements are present, and  
6973 the IDREF references an element containing some *AnchorCertificateRef*, then an opportunity  
6974 exists to check suitability of one *Party*'s PKI trust of the certificates used in the  
6975 *TransportSecurityProtocol*. For example, by resolving the IDREF value in  
6976 *TransportSender/TransportClientSecurity/ClientCertificateRef/@certId*, we can obtain the  
6977 proposed client certificate to use for client-side authentication. By resolving the IDREFs from  
6978 the *AnchorCertificateRef*, we become able to determine whether the proposed client certificate  
6979 will "chain to a trusted root" on the server side's PKI. Similar remarks apply to checks on the  
6980 validity of a server certificate found by resolving  
6981 *TransportReceiver/TransportServerSecurity/ServerCertificateRef*. This server certificate can  
6982 be checked against the CA trust anchors that are found by resolving  
6983 *TransportSender/TransportClientSecurity/ServerSecurityDetailsRef/@securityId*, and finding  
6984 CA certificates (or CA certificate chains) in the *KeyInfo* elements under the *Certificate* element  
6985 obtained by resolving the IDREF found in *AnchorCertificateRef@certId*.

6986  
6987 When matches exist for the correlative *Transport* components, we then have discovered an  
6988 interoperable solution at the transport level. If not, no *CPA* will be available, and a gap has been  
6989 identified that will need to be remedied by whatever exception handling procedures are in place.  
6990 Let us next consider other capabilities that need to match for "thicker" interoperable solutions.

#### 6991 6992 **E.5.2.2 Checking BusinessTransactionCharacteristics and DeliveryChannel** 6993 **MessagingCharacteristics**

6994 Under each of the correlative action bindings, there is a child element of *DeliveryChannel*,  
6995 *MessagingCharacteristics* that has several attributes important in *CPA* formation tasks. The  
6996 attributes having wider implications are *syncReplyMode*, *ackRequested*, and  
6997 *ackSignatureRequested*; for the *duplicateElimination* and *actor* attributes, compatibility exists  
6998 when the attributes that are found under the *CanSend* and *CanReceive DeliveryChannels* have  
6999 the same values. As the element's name implies, all of these *DeliveryChannel* features pertain to  
7000 the messaging layer.

7001  
7002 In addition, *BusinessTransactionCharacteristics*, found under *ThisPartyActionBinding*,  
7003 contains attributes reflecting a variety of features pertaining to desired security and business  
7004 transaction properties that are to be implemented by the agreed upon *DeliveryChannels*. These  
7005 properties may have implications on what capabilities are needed within more detailed  
7006 components of the *DeliveryChannel* elements, such as in the *Packaging* element. When using a  
7007 *BPSS* process specification, these properties may be specified within the *BusinessTransaction*.  
7008 The properties of the *BusinessTransactionCharacteristics* element are, however, the ones that  
7009 will be operative in the implementation of the *BusinessTransaction*, and may override the  
7010 specified values found in the *BPSS* Process specification. Because the properties are diverse, the  
7011 details that implement the properties can be spread over other elements referenced within the  
7012 *DeliveryChannel* elements.

7013  
7014 These attributes apply to either a *Request* or *Response* delivery channel, but can impact either the  
7015 *Sender* or *Receiver* (or both) in a channel. In addition, the attributes governing  
7016 acknowledgments, for example, qualify the interrelation of *DeliveryChannel* elements by  
7017 specifying behavior that is to occur that qualifies the contents of a return message.

7018  
7019 The most basic test for compatibility for any of the attributes in either *MessagingCharacteristics*  
7020 or *BusinessTransactionCharacteristics* is that the attributes are equal in the sending party's  
7021 *DeliveryChannel* referenced by *CanSend/ThisPartyActionBinding/ChannelId* and in the  
7022 receiving party's *DeliveryChannel* referenced by  
7023 *CanReceive/ThisPartyActionBinding/ChannelId*. If they are unequal, and all Bindings have  
7024 been examined on both sides, a draft *CPA* will represent a compromise to some common set with  
7025 respect to the functionality represented by the attributes.

7026  
7027 In the following discussions, we will consider many of the attributes in the two *Characteristics*  
7028 elements, and relate them to additional underlying implementational details, one of which is  
7029 **Packaging**.

7030  
7031 From a high level, basic agreement in *packaging* is a matter of compatibility of the generated  
7032 *packaging* on the sending side with the parsed packaging on the receiving side. The basic  
7033 packaging check is, therefore, checking packaging compatibility under the *CanSend* element of a  
7034 sender action with the packaging under the *CanReceive* element of that same action under the  
7035 receiver side.

7036  
7037 For efficiency, representation of capabilities of parsing/handling packaging can make use of both  
7038 wildcards and repetition, and as needed these capabilities can also express open data formatting  
7039 used on the generating side. For example, consider the **SimplePart**:

```
7040 <tp:SimplePart tp:id="IWild" tp:mimetype="*/*/"/>
```

7041  
7042 By wildcarding *mimetype* values, we represent our capability of accepting any data, and would  
7043 match any specific MIME type. Also, consider a **Constituent** appearing within a **Composite**:

```
7044 <tp:Constituent tp:idref="MsgHdr"/>  
7045 <tp:Constituent minOccurs="0" maxOccurs="10" tp:idref="IWild"/>
```

7046  
7047 This notation serves to capture the capability of handling any number of arbitrary MIME  
7048 bodyparts within the **Composite** being defined. A Packaging capability such as this would  
7049 obviously match numerous more specific generated *Packaging* schemes, as well as matching  
7050 literally with a scheme of the same generality.

7051  
7052 Certain more complex checks are needed for more complicated packaging options pertaining to  
7053 *syncReplyMode*. These are discussed in the following.

#### 7054 ***syncReplyMode***

7055  
7056 The ***syncReplyMode*** has a value other than "none" to indicate what parts of a message should be  
7057 returned in the *Reply* of a transport capable of synchronous operation, such as HTTP. (We here

7060 use “synchronous” to mean “on the same TCP connection,” which is one use of this term. We do  
7061 not specify any waiting, notification, or blocking behavior on processes or threads that are  
7062 involved, though presumably there is some computational activity that maintains the connection  
7063 state and is above the TCP and socket layers.)

7064  
7065 The possible implementations pertaining to various values of the *syncReplyModes* are numerous,  
7066 but we will try to indicate at least the main factors that are involved.

7067  
7068 As will be seen, the *Packaging* element is important in specifying implementation details and  
7069 compatibilities. But, because business level signals may be involved, other action bindings may  
7070 need examination in addition to the already selected bindings for the *Request* and *Response*.  
7071 Also, the values of *TransportReceiver/Endpoint/@type* might need checking when producing  
7072 draft CPAs.

7073  
7074 Let us first begin with the cases in which *Responses*, *Message Service Handler Signals* and  
7075 *Business Signals* return in some combination of a synchronous reply and other asynchronous  
7076 message(s). These various combinations will be discussed for the *syncReplyMode* values:  
7077 "mshSignalsOnly," "signalsOnly," "responseOnly," and "signalsAndResponse."

7078  
7079 By convention, synchronous replies are represented by subordinating *CanSend* or *CanReceive*  
7080 elements under the *CanReceive* or *CanSend* elements that represent the initial *Request* binding  
7081 capabilities. For representing asynchronous Requests, Replies, or Signals, the *CanSend* or  
7082 *CanReceive* elements are all siblings and directly subordinate to the *ServiceBinding*. Therefore,  
7083 both asynchronous and synchronous capabilities can be grouped under a *ServiceBinding* in a  
7084 CPP, and can still be unambiguously distinguished. In principle, increasing subordination  
7085 (nesting) can indicate patterns of dialog more elaborate than *Request* and *Response*. Few use  
7086 cases for this functionality are common at the time of this writing.

7087  
7088 *mshSignalsOnly*

7089 The Request sender's *DeliveryChannel* (referenced by  
7090 *CanSend/ThisPartyActionBinding/ChannelId*) and the Request receiver's *DeliveryChannel*  
7091 (referenced by *CanReceive/ThisPartyActionBinding/ChannelId*) both should have  
7092 *MessagingCharacteristics/@syncReplyMode* value of *mshSignalsOnly*.

7093  
7094 While a Party can explicitly identify a *DeliveryChannel* for the SOAP envelope with subordinate  
7095 *CanSend* and *CanReceive* elements, and with them specialized bindings, these are typically  
7096 omitted for ebXML Messaging software. It is presumed that each side can process a synchronous  
7097 reply constructed in accordance with ebXML Messaging. The *DeliveryChannel* representation  
7098 mechanism here serves as a placeholder for capturing other Messaging Signal protocols that  
7099 might emerge.

7100  
7101 Currently acknowledgments and signed acknowledgments, along with errors, are the primary  
7102 MSH signals that are included in the SOAP envelope. If Company A set  
7103 *syncReplyMode* to *mshSignalsOnly*, then Company B's correlative  
7104 *CanReceive/ThisPartyActionBinding/@packageId* should contain a nested

7105 ***CanSend/ThisPartyActionBinding/@packageId*** for a message without any business payload or  
7106 signals. In addition, the ***CanSend/ThisPartyActionBinding/@packageId*** of Company B's  
7107 ***Response*** should resolve to packaging format capable of returning the ***Response*** ( and possibly  
7108 other constituents) asynchronously. The compatibility of the ***DeliveryChannel*** elements can be  
7109 checked, as can the capability of Company A to receive that Response payload, the Signal  
7110 payload(s), or Responses bundled with signals as specified by the packaging formats that are  
7111 referenced through the relevant ***ThisPartyActionBindings*** element's ***packageId*** attribute values.

7112

### 7113 ***signalsOnly***

7114 The Request sender's ***DeliveryChannel*** (referenced by its  
7115 ***CanSend/ThisPartyActionBinding/ChannelId***) and the Request receiver's ***DeliveryChannel***  
7116 (referenced by its ***CanReceive/ThisPartyActionBinding/ChannelId***) both should have  
7117 ***MessagingCharacteristics/@syncReplyMode*** value of ***signalsOnly***.

7118

7119 If Company A sets ***syncReplyMode*** to ***signalsOnly***, then under Company B's correlative  
7120 ***CanReceive*** element, there should be a nested ***CanSend/ThisPartyActionBinding*** whose  
7121 ***packageId*** attribute's value resolves to a packaging format appropriate for Signals. For the  
7122 ***CanSend/ThisPartyActionBinding/@packageId*** associated with Company B's business level  
7123 ***Response***, the attribute IDREF value should resolve to a packaging format capable of returning  
7124 payloads and that omits business signals. This ***CanSend*** element will be a direct child of  
7125 ***ServiceBinding***, a placement representing its asynchronous character. The original requesting  
7126 party will need to have a ***CanReceive/ThisPartyActionBinding*** that is compatible with the  
7127 responding party, and that is a direct child of its ***ServiceBinding*** element.

7128

7129 Using subordinate ***CanSend*** and subordinate ***CanReceive*** elements can be useful if the  
7130 ***DeliveryChannel*** details for Exception signals differ from those specified for Request and  
7131 Response. Signal bindings, for example, may differ by omitting ***ackRequested***, or possibly one  
7132 of the security features (digital enveloping or non-repudiation of receipt) that are used for  
7133 Requests or Responses. Just as with other tests on Requests and Responses, there can be checks  
7134 for compatibility in ***Packaging***, ***DocExchange***, ***MessagingCharacteristics***, or  
7135 ***BusinessTransactionCharacteristics*** referred to in the correlative subordinate ***CanSend*** and  
7136 ***CanReceive DeliveryChannels***.

7137

### 7138 ***responseOnly***

7139 The Request sender's ***DeliveryChannel*** (referenced by  
7140 ***CanSend/ThisPartyActionBinding/ChannelId***) and the Request receiver's ***DeliveryChannel***  
7141 (referenced by ***CanReceive/ThisPartyActionBinding/ChannelId***) both should have  
7142 ***MessagingCharacteristics/@syncReplyMode*** value of ***responseOnly***.

7143

7144 If Company A sets ***syncReplyMode*** to ***responseOnly***, the  
7145 ***CanSend/ThisPartyActionBinding/@packageId*** of Company B's response should resolve to a  
7146 packaging format capable of returning payloads, but omitting business signals. The  
7147 ***CanSend/ThisPartyActionBinding*** element will be included as a child of the ***CanReceive***  
7148 element so the responder can indicate that it is a synchronous response.

7149

7150 There should be an independent way to return business level error signals. So, there should be a



7151 ***ThisPartyActionBinding*** for any Signal payload announced, and these bindings should be at the  
7152 direct child of ***ServiceBinding*** level to represent their asynchronous flavor.

7153

7154 It is not too likely that ***ReceiptAcknowledgment*** and similar signals will be used when a response  
7155 is returned synchronously. The motivation for using these signals is indicating positive forward  
7156 progress, and this motivation will be undermined when a Response is returned directly.

7157

7158 For the ***responseOnly*** case, including subordinate ***CanSend/ThisPartyActionBinding*** and  
7159 ***CanReceive/ThisPartyActionBinding***, means that there can be checks for compatibility in  
7160 ***Packaging, DocExchange, MessagingCharacteristics, or BusinessTransactionCharacteristics***.  
7161 The ***syncReplyMode*** and ***ackRequested*** attributes here should be carefully considered because a  
7162 ***mshSignalsOnly*** value here would mean that another round of synchronous messaging will need  
7163 to occur on the same connection. Incidentally, for ***Transport*** elements referenced under  
7164 subordinate bindings, there need not be any ***Endpoint*** elements. If there are ***Endpoint*** elements,  
7165 they may be ignored.

7166

#### 7167 ***signalsAndResponse***

7168 The Request sender's ***DeliveryChannel*** (referenced by  
7169 ***CanSend/ThisPartyActionBinding/ChannelId***) and the Request receiver's ***DeliveryChannel***  
7170 (referenced by ***CanReceive/ThisPartyActionBinding/ChannelId***) both should have  
7171 ***MessagingCharacteristics/@syncReplyMode*** value of ***signalsAndResponse***.

7172

7173 If Company A sets ***syncReplyMode*** to ***signalsAndResponse***, the  
7174 ***CanSend/ThisPartyActionBinding*** of Company B's response should be subordinate to Company  
7175 B's ***CanReceive*** element. The packaging format that is referenced should be capable of  
7176 returning payloads and signals bundled together. If no asynchronous bindings exist for error  
7177 signals, this will be the only defined ***DeliveryChannel*** agreed to for all aspects of message  
7178 exchange for the business transaction. However, it is likely that an asynchronous binding would  
7179 normally be provided to send Exception signals.

7180

#### 7181 ***ackRequested and ackSignatureRequested***

7182 Checks on the ***ackRequested*** and ***ackSignatureRequested*** attributes within correlative  
7183 ***DeliveryChannels*** (that is, correlative because referenced under one action's ***CanSend*** and  
7184 ***CanReceive*** elements) are primarily to see that the values of the corresponding attributes are the  
7185 same.

7186

7187 However, there are some interactions of these attributes with other information items that need to  
7188 be mentioned.

7189

7190 The principal use of the ***ackRequested*** attribute is within reliable messaging configurations. If  
7191 reliable messaging is to be configured, then checks on agreement in the correlative  
7192 ***ReliableMessaging*** elements as found under ***DocExchange/ebXMLSenderBinding*** and  
7193 ***DocExchange/ebXMLReceiverBinding*** are in order. Also, the value of the  
7194 ***duplicateElimination*** attribute of ***MessagingCharacteristics*** should be checked for agreement.  
7195 Draft CPAs may be formed by deliberately aligning values that are not equal along some of these  
7196 dimensions. Downgrading may provide draft CPAs most likely to gain acceptance; so, for

7197 example, if *duplicateElimination* is false on the receiving side, aligning it to false on the sending  
7198 side is most likely to produce a draft that succeeds.

7199  
7200 The additional function of *ackSignatureRequested* is that it provides a “thin” implementation for  
7201 *non-repudiation of receipt*. The basic check is for equality of attribute value, but additional  
7202 constraints may need test and alignment. If no signal capable of implementing *non-repudiation*  
7203 *of receipt* is found under the *ServiceBinding*, then having an “always” value for  
7204 *ackSignatureRequested* suggests aligning the *BusinessTransactionCharacteristics* attributes,  
7205 *isNonRepudiationReceiptRequired*, to be true. However, if this is done, care should be taken to  
7206 check that the *BusinessTransactionCharacteristics* attribute *isIntelligibleCheckRequired* is  
7207 false. This is because the messaging implementation only deals with receipt in the sense of  
7208 having received a byte stream off the wire (and persisting it so that it is available for further  
7209 processing). It is not safe to presume that any syntactical or semantic checks on the data were  
7210 performed.

### 7211 **E.5.2.3 DocExchange Checks for BusinessTransactionCharacteristics**

7212 When using *CPPs* and *CPAs* with ebXML Messaging, which is the most likely early deployment  
7213 situation, there exists an opportunity to check agreement on *BusinessTransactionCharacteristics*  
7214 attributes:

7215  
7216 The following three attributes need to have equal values in the bindings for a Request or for a  
7217 Response. No further discussion will be provided in this appendix on these “deadlines,” except to  
7218 say that a sophisticated proposed *CPA* generation tool might check on the coherence of the  
7219 values chosen here with values for reliable messaging parameters, existence of compatible  
7220 ReceiptAcknowledgment or AcceptanceAcknowledgment bindings, and consistency with  
7221 syncReplyMode internal configuration.

```
7222  
7223  
7224 <attribute name="timeToAcknowledgeReceipt" type="duration"/>  
7225 <attribute name="timeToAcknowledgeAcceptance" type="duration"/>  
7226 <attribute name="timeToPerform" type="duration"/>  
7227
```

7228 The remaining attributes involve a number of security related issues and will be the focus of the  
7229 remaining discussion of *BusinessTransactionCharacteristics* attributes:

```
7230  
7231 <attribute name="isNonRepudiationRequired" type="boolean"/>  
7232 <attribute name="isNonRepudiationReceiptRequired" type="boolean"/>  
7233 <attribute name="isIntelligibleCheckRequired" type="boolean"/>  
7234 <attribute name="isAuthenticated" type="tns:persistenceLevel.type"/>  
7235 <attribute name="isTamperProof" type="tns:persistenceLevel.type"/>  
7236 <attribute name="isAuthorizationRequired" type="boolean"/>  
7237 <attribute name="isConfidential" type="tns:persistenceLevel.type"/>  
7238 <attribute name="isSecureTransportRequired" type="boolean"/>  
7239
```

7240 Here, the basic test is that for correlative *DeliveryChannels*, the corresponding attributes have  
7241 the same values. Again there are some interaction aspects with parts of the *DeliveryChannel* that  
7242 motivate making some additional checks.

7243  
7244 Previously, when discussing the *MessagingCharacteristics* attribute *ackSignatureRequested*, it  
7245 was pointed out that the messaging implementation provides thin support for holding

7246 ***isNonRepudiationReceiptRequired*** true provided that the attribute ***isIntelligibleCheckRequired***  
7247 is false. When both are true, then there should exist a business signal with compatible ***Packaging***  
7248 and ***DeliveryChannel*** values. If the signal has been independently described within  
7249 asynchronous ***CanSend*** and ***CanReceive*** elements, knowing the signal name (such as,  
7250 “ReceiptAcknowledgment”) may support a relatively simple search and test. However, if  
7251 synchronous transports are involved, some filters using ***syncReplyModes*** may be needed to  
7252 discover an underlying support for a “thick” implementation of *non-repudiation of receipt*.  
7253

7254 When non-repudiation of receipt is implemented by a business signal, then checks on signing  
7255 certificate validity can involve the ***CollaborationRole/ApplicationCertificateRef*** and the  
7256 ***CollaborationRole/ApplicationSecurityDetailsRef***, that provides a reference to the  
7257 ***SecurityDetails*** element containing the list of ***TrustAnchors***. The certificate from the side  
7258 signing the ReceiptAcknowledgment would be checked against the certificates referred to by the  
7259 ***AnchorCertificateRef*** under ***TrustAnchors***.  
7260

7261 The business signal will sometimes be conveyed as part of a message. It remains true that the  
7262 message itself will still be sent through a MSH, and that the MSH can also sign the message  
7263 using the certificate found by resolving the IDREF found at  
7264 ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId***.  
7265

7266 If a particular software component implements both MSH functionality and business level  
7267 security functionality, it is possible that the same certificate may be pointed to by  
7268 ***ApplicationCertificateRef*** and ***SigningCertificateRef/@certId***. In other words, the distinction  
7269 between MSH level signing and application level signing is a logical one, and may not  
7270 correspond with software component boundaries. Because the MSH signature is over the  
7271 message, the message signature may be over an application level signature. While this may be  
7272 redundant for some system configurations, protocols may require both signatures to exist over  
7273 the different regions.  
7274

7275 Failure to validate a certificate may not prevent formation of a draft *CPA*. First, the sender’s signing  
7276 certificate can be a self-signed certificate. If so, a reference to this self-signed certificate may be  
7277 added to the receiver’s ***TrustAnchors/AnchorCertificateRef*** list. This proposal amounts to  
7278 proposing to agree to a direct trust model, rather than a hierarchical model involving Certificate  
7279 Authorities. Second, a proposal to add a trusted root may be made, again by appropriate revision of  
7280 the ***TrustAnchors***.  
7281

7282 When non-repudiation of receipt is implemented by the Messaging layer, the checks on PKI  
7283 make use of elements under ***DocExchange***.  
7284

7285 ***isNonRepudiationRequired***  
7286 ***isAuthenticated***  
7287 ***isAuthorizationRequired***  
7288 ***isTamperProof***  
7289

7290 The ideas of authentication, authorization, nonrepudiation and being “tamper proof” may be very  
7291 distinct as business level concepts, yet the implementation of these factors tend to use very

7292 similar technologies. Actually, prevention of tampering is not literally implemented. Instead,  
7293 means are provided for detecting that tampering (or some accidental garbling) has occurred.  
7294 Likewise, implementations of authorization usually are provided by implementations of access  
7295 control (permitting or prohibiting a user in a role making use of a resource) and presentation of a  
7296 token or credential to gain access, which may involve authentication as an initial step!  
7297 Nonrepudiation may build on all the previous functions, plus retaining information for supplying  
7298 presumptive evidence of origination at some later time.

7299  
7300 When checking whether *isNonRepudiationRequired* can be set to True for both parties, check  
7301 whether the signing certificate will be counted as valid at the receiver.

7302 The IDREF reference to the signing certificate is found in  
7303 *DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId*.  
7304 The referenced certificate should be checked for validity with respect to the trust anchors  
7305 obtained from *TrustAnchors/AnchorCertificateRef* elements under the *SecurityDetails*  
7306 element referenced by the IDREF at  
7307 *DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation/SigningSecurityDetailsRef/*  
7308 *@securityId*.

7309  
7310 As previously noted, failure to validate a certificate does not prevent constructing a draft CPA.  
7311 Either self-signed certificates or new trust anchors can be added to align the trust model on one  
7312 side with the other side's certificate.

7313  
7314 In addition to checking the interoperability of the PKI infrastructures, checks on compatibility of  
7315 values in the other attributes in  
7316 *DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation* and in  
7317 *DocExchange/ebXMLSenderBinding/SenderNonRepudiation* can be made.  
7318 *NonRepudiationProtocol*, *HashFunction*, and *SignatureAlgorithm* values may be compatible  
7319 even when not equal if knowledge of the protocol requirements allows fallback to a mandatory to  
7320 implement value. So values here can be found equal, aligned, or negotiated to reach an  
7321 agreement.

7322  
7323 If *isNonRepudiationRequired* is True, the *isAuthenticated* and *isTamperProof* should also be  
7324 True. This is because in implementing *isNonRepudiationRequired* by means of a digital  
7325 signature, both authentication (with respect to the identity associated with the signing certificate)  
7326 and tamper detection (with respect to the cryptographic hash of the signature) will be  
7327 implemented as well. The converses need not be true because authentication and tamper  
7328 detection might be accomplished without archiving information needed to support claims of  
7329 nonrepudiation.

7330  
7331 *isConfidential*  
7332 *isSecureTransportRequired*

7333  
7334 The *isConfidential* and *isSecureTransportRequired* attributes indicate properties variously  
7335 distributed among levels of the application-to-application sending/receiving stacks.  
7336 *isConfidential* has possible values of "none", "transient", "persistent", and "transient-and-  
7337 persistent". If *isSecureTransportRequired* is true, there should be compatible

7338 **TransportSecurity**, and **isConfidential** needs to have either a “transient” or a “transient-and-  
7339 persistent” value for consistency. The “persistent” or “transient-and-persistent” values indicate  
7340 that some digital enveloping function is present.

7341  
7342 ebXML Messaging version 2.0 does not have an “official” implementation for digital envelopes,  
7343 and refers to the future XML Encryption specification as its intended direction for that function.  
7344 However, the XML Encryption specification is now a candidate recommendation, and is suitable  
7345 for preliminary implementation.

7346  
7347 Within the *CPA*, the **DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope** and  
7348 **DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope** can provide configuration  
7349 details pertaining to security in accordance with [XMLENC]. Use of XML Encryption also will  
7350 normally show up in the value of **DigitalEnvelopeProtocol**, and can also appear within a  
7351 **NamespaceSupported** element within **Packaging**.

7352  
7353 Currently, [ebMS] has only indicated a direction to eventually use XML Encryption, but has not  
7354 mandated any digital envelope protocol. Digital enveloping may be done at the “application  
7355 level,” and will show up under MIME types within the **Packaging** element. PKI matching will  
7356 make use of certificates supplied in **ApplicationCertificateRef** and  
7357 **ApplicationSecurityDetailsRef**. If other protocols are to be used, it would be safest to use  
7358 extensions to the content model of **DocExchange**, such as, **XXXSenderBinding** and  
7359 **XXXReceiverBinding**, and follow the pattern of the ebXML content models for **DocExchange**.  
7360 Future versions of this specification intend to make these extension semantics easier to use  
7361 interoperably; currently, the extensions would be a multilateral extension within some trading  
7362 community.

7363  
7364 When checking whether **isConfidential** can be set to “persistent” or “transient-and-persistent”  
7365 for both parties, check whether the key exchange certificate will be counted as valid at the  
7366 sender. The IDREF reference to the **SecurityDetails** element is found in  
7367 **DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope/EncryptionSecurityDetailsRef/@**  
7368 **securityId**. The trust anchor certificates obtained from **TrustAnchors/AnchorCertificateRef**  
7369 elements under the **SecurityDetails** element will be used to test that the certificate referenced by  
7370 **DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@c**  
7371 **ertId** validates at the sender side.

7372  
7373 As previously noted, failure to validate a certificate does not prevent constructing a draft CPA.  
7374 Either self-signed certificates or new trust anchors can be added to align the trust model on one  
7375 side with the other side’s certificate.

7376  
7377 In addition to the PKI related checks and alignments, the elements **EncryptionAlgorithm** and  
7378 **DigitalEnvelopeProtocol** should be checked for equality (or compatibility) and, if not  
7379 compatible or equal, aligned to values that would work for an initial version of a proposed CPA.  
7380 Preferences and alignment of these elements can be achieved in a subsequent Negotiation phase.

7381  
7382 Finally, it is possible that one side’s DigitalEnvelope will be modeled using either the  
7383 **DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope** and

7384 ***DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope***, while the other side uses only  
 7385 ***Packaging*** to indicate use of, for example, S/MIME Digital Envelopes, because it receives an  
 7386 already enveloped payload from an application. In such a case, the PKI certificate  
 7387 validationcheck could require checking that a certificate described by  
 7388 ***DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@c***  
 7389 ***ertId*** validates against the ***TrustAnchors*** found by resolving  
 7390 ***CollaborationRole/ApplicationSecurityDetailsRef***. This complication arises from the possibility  
 7391 that digital enveloping functionality can be spread over quite distinct portions of the stack in  
 7392 different software installations.  
 7393

## 7394 **E.6 CPA Formation: Technical Details**

7395 When assembling a draft *CPA* from matching portions of two *CPPs*' ***PartyInfo*** elements, some  
 7396 additional constraints need to be observed.

7397

7398 First, as mentioned in section 9.11.1, software for producing draft CPAs needs to guarantee that  
 7399 ID values in one *CPP* are distinct from ID values in the other *CPP* so that no IDREF references  
 7400 collide when the *CPPs* are merged. The following ID values are potentially subject to collision:  
 7401

7402 ***Certificates***  
 7403 ***SecurityDetails***  
 7404 ***SimplePart***  
 7405 ***Packaging***  
 7406 ***DocExchange***  
 7407 ***Transport***  
 7408 ***DeliveryChannel***  
 7409 ***ThisPartyActionBinding***

7410

7411 There are elements and complex type definitions containing IDREFs. Also some elements have  
 7412 attributes with IDREF values. These are:  
 7413

7414 ***PartyInfo***  
 7415 ***ActionBinding.type***  
 7416 ***ThisPartyActionBinding***  
 7417 ***OtherPartyActionBinding***  
 7418 ***OverrideMSHActionBinding***  
 7419 ***ChannelId***  
 7420 ***DeliveryChannel***  
 7421 ***Constituent***  
 7422 ***CertificateRef.type***  
 7423 ***AnchorCertificateRef***  
 7424 ***ApplicationCertificateRef***  
 7425 ***ClientCertificateRef***  
 7426 ***ServerCertificateRef***  
 7427 ***SigningCertificateRef***  
 7428 ***EncryptionCertificateRef***

7429            *CertificateRef*  
7430            *SecurityDetailsRef.type*

7431  
7432        Second, when the *CanSend* and *CanReceive* binding information has been found to match  
7433        (equal, correspond with, or be compatible with) the binding information under the other Party's  
7434        *CanReceive* and *CanSend* elements, the IDREF references for the *OtherPartyActionBinding*  
7435        are filled out in the CPA.

7436  
7437        Third, for CPAs that are signed, the implementer is advised to review section 9.9.1.1 when using  
7438        [XMLDSIG] for the signature technique. A proposed CPA need not have a signature.

7439  
7440        Fourth, when a CPA is composed from two CPPs, see section 8.8 in which it stated that all  
7441        *Comment* elements from both CPPs SHALL be included in the CPA unless agreed to otherwise.

7442  
7443        Fifth, several tests on CPA validity could be conducted on draft CPAs, but these tests are more  
7444        critical for a negotiated CPA that is to be deployed and imported into run-time software  
7445        components.

7446  
7447        1. Expiration: Certificates used in signing a CPA can be checked to verify that they do not expire  
7448        before the CPA expires, as given in the *End* element.

7449  
7450        2. Certificate expiration: If a CPA lifetime exceeds the lifetime of certificates accepted for use in  
7451        signing, key exchange or other security functions, then it would be advisable to make ds:KeyInfo  
7452        refer to certificates, rather than to include them within the element by value.

7453  
7454        3. Process-Specification references can be checked in accordance with the provisions of section  
7455        8.4.4 and its subsections.

7456  
7457        Finally, a CPA has several elements whose values are not typically derived from either CPPs  
7458        (and can need checking when using a CPA template as the basis for a draft CPA.) The *Status*,  
7459        *Start*, *End*, and possibly a *ConversationConstraints* element need to be added. The attributes,

7460  
7461            *CollaborationProtocolAgreement/@cpaid*,  
7462            *CollaborationProtocolAgreement/@version*,  
7463            *CollaborationProtocolAgreement/Status@value*,  
7464            *CollaborationProtocolAgreement/ConversationConstrain@invocationLimit*, and  
7465            *CollaborationProtocolAgreement/ConversationConstraint@concurrentConversations*,

7466  
7467        can also be supplied values as needed.

7468 Appendix F Correspondence Between CPA and ebXML  
7469 Messaging Parameters (Normative)

7470 The following table shows the correspondence between elements used in the ebXML Messaging  
7471 Service message header and their counterparts in the CPA.  
7472  
7473

<b>Message Header Element / Attribute</b>	<b>Corresponding CPA Element / Attribute</b>
<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



	<i>Manifest</i> SHOULD correspond to a <i>SimplePart</i> that is referenced from one of the <i>CompositeList</i> elements under <i>Packaging</i>
<i>xlink:role</i> attribute in <i>Reference</i> element	<i>xlink:role</i> attribute in <i>SimplePart</i> element
<i>AckRequested</i> element	<i>ackRequested</i> attribute in <i>MessagingCharacteristics</i> element; an <i>AckRequested</i> element is included in the SOAP Header if the <i>ackRequested</i> attribute is set to “always”; if it is set to “perMessage”, input passed to the MSI is to be used to determine if an <i>AckRequested</i> element needs to be included; likewise, the signed attribute under <i>AckRequested</i> will be appropriately set based on the <i>ackSignatureRequested</i> attribute and possibly determined by input passed to the MSI
<i>MessageOrder</i> element	<i>messageOrderSemantics</i> attribute in <i>ReliableMessaging</i> element; the <i>MessageOrder</i> element will be present if the <i>AckRequested</i> element is present, and if the <i>messageOrderSemantics</i> attribute in the <i>ReliableMessaging</i> element is set to "Guaranteed"
<i>ds:Signature</i> element	<i>ds:Signature</i> will be present in the SOAP Header if the <i>isNonRepudiationRequired</i> attribute in the <i>BusinessTransactionCharacteristics</i> element is set to “true”; the relevant parameters for constructing the signature can be obtained from the <i>SenderNonRepudiation</i> and <i>ReceiverNonRepudiation</i> elements

7474

7475

7476

7477

7478

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
<i>Retries</i> (not in Message Header) but used to govern Reliable Messaging behavior in sender	<i>Retries</i> element (under <i>ReliableMessaging</i> element)
<i>RetryInterval</i> (not in Message Header) but used to govern Reliable Messaging behavior in sender	<i>RetryInterval</i> element (under <i>ReliableMessaging</i> element)
<i>PersistDuration</i> (not in Message Header) but used to govern Reliable Messaging behavior in receiver	<i>PersistDuration</i> element (under <i>ebXMLReceiverBinding</i> element)
<i>Endpoint</i> (not in Message Header) but used for sending SOAP message	<i>Endpoint</i> element (under <i>TransportReceiver</i> ); the type of message

	being sent <b>MUST</b> be passed in to the MSI; an appropriate endpoint can then be selected from among the <b>Endpoints</b> included under the <b>TransportReceiver</b> element
Use <b>Service &amp; Action</b> to determine the corresponding <b>DeliveryChannel</b>	<b>DeliveryChannel</b>
Use <b>ReceiverDigitalEnvelope</b> to determine the encryption algorithm and key	<b>ReceiverDigitalEnvelope</b>
Use <b>SenderNonRepudiation</b> to determine signing certificate(s) and <b>ReceiverNonRepudiation</b> to determine the trust anchors and security policy to apply to the signing certificate	<b>SenderNonRepudiation</b> and <b>ReceiverNonRepudiation</b>
Use <b>Packaging</b> to determine how payload containers ought to be encapsulated. Also use <b>Packaging</b> to determine how an individual SimplePart ought to be extracted and validated against its schema	<b>Packaging</b>
Use <b>TransportClientSecurity</b> and <b>TransportServerSecurity</b> to determine certificates to be used by server and client for authentication purposes	<b>TransportClientSecurity</b> and <b>TransportServerSecurity</b>
Use the <b>DeliveryChannel</b> identified by <b>defaultMshChannelId</b> for standalone MSH level messages like Acknowledgment, Error, StatusRequest, StatusResponse, Ping, Pong, unless overridden by <b>OverrideMshActionBinding</b>	<b>defaultMshChannelId</b> attribute in <b>PartyInfo</b> element, and <b>OverrideMshActionBinding</b>

## 7479 Appendix G Glossary of Terms

7480

Term	Definition
AGREEMENT	An arrangement between two partners that specifies in advance the conditions under which they will trade (terms of shipment, terms of payment, collaboration protocols, etc.) An agreement does not imply specific economic commitments.
APPLICATION	Software above the level of the MSH that implements a Service by processing one or more of the Messages in the Document Exchanges associated with the Service.
AUTHORIZATION	A right or a permission that is granted to a system entity to access a system resource.
BUSINESS ACTIVITY	A business activity is used to represent the state of the business process of one of the partners. For instance the requester is either in the state of sending the request, in the state of waiting for the response, or in the state of receiving.
BUSINESS COLLABORATION	An activity conducted between two or more parties for the purpose of achieving a specified outcome.
BUSINESS DOCUMENT	The set of information components that are interchanged as part of a business activity.
BUSINESS PARTNER	An entity that engages in business transactions with another business partner(s).
BUSINESS PROCESS	The means by which one or more activities are accomplished in operating business practices.
BUSINESS PROCESS SPECIFICATION SCHEMA	Defines the necessary set of elements to specify run-time aspects and configuration parameters to drive the partners' systems used in the collaboration. The goal of the BP Specification Schema is to provide the bridge between the eBusiness process modeling and specification of eBusiness software components.
BUSINESS TRANSACTION	A business transaction is a logical unit of business conducted by two or more parties that generates a computable success or failure state. The community, the partners, and the process, are all in a definable, and self-reliant state prior to the business transaction, and in a new definable, and self-reliant state after the business transaction. In other words if you are still 'waiting' for your business partner's response or reaction, the business transaction has not completed.
CLIENT	Software that initiates a connection with a <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.
--------------------------------------	---

7481