



Creating A Single Global Electronic Market

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

ebXML Business Process Specification Schema Version 0.99

Context/Metamodel Group
of the CC/BP Joint Delivery Team

03/19/2001

1 Status of this Document

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34

This document is a working DRAFT for the *eBusiness* community. Distribution of this document is unlimited. This document will go through the formal *Quality Review* Process as defined by the *ebXML Requirements Document*. The formatting for this document is based on the Internet Society's Standard RFC format.

This version:

EbXML_BPschema_0.99

Latest version:

EbXML_BPschema_0.99

Previous version:

EbXML_BPschema_0.90

ebXML Business Process Specification Schema

Copyright © ebXML 2000 & 2001. All Rights Reserved.

35 **2 ebXML BP/CoreComponents metamodel participants**

36 We would like to recognize the following for their significant participation to the development of
37 this document.

38

39 Team Lead:

40 Paul Levine, Telcordia

41

42 Editors:

43 Jim Clark, I.C.O.T. - previously Edifecs: (Transaction Semantics)

44 Cory Casanave, Data Access Technologies: (UML model)

45 Kurt Kanaskie, Lucent Technologies: (DTD and Examples)

46 Betty Harvey, Electronic Commerce Connection: (DTD documentation)

47 Jamie Clark, Spolin Silverman & Cohen LLP: (Legal aspects)

48 Karsten Riemer, Sun Microsystems: (Overall Document)

49

50 Participants:

51 Antoine Lonjon, Mega

52 J.J. Dubray, Excelon

53 Bob Haugen, Logistical Software

54 Bill McCarthy, Michigan State University

55 Paul Levine, Telcordia

56 Brian Hayes, CommerceOne

57 Nita Sharma, Netfish

58 David Welsh, Nordstrom

59 Antonio Carrasco, Data Access Technologies

60 Neal Smith, Chevron

61

62 3 Table of Contents

63	1	Status of this Document	i
64	2	ebXML BP/CoreComponents metamodel participants	ii
65	3	Table of Contents	iii
66	4	Introduction	v
67		Executive Summary	v
68	4.1	Summary of Contents of Document.....	1
69	4.2	Audience.....	1
70	4.3	Related Documents	1
71	4.4	Prerequisites	1
72	5	Design Objectives	2
73	5.1	Goals/Objectives/Requirements/Problem Description.....	2
74	5.2	Caveats and Assumptions.....	2
75	5.2.1	Relationship between Specification Schema and UMM.....	2
76	6	System Overview	4
77		UML Specification Schema	6
78		DTD Specification Schema.....	6
79		Business Process Interaction Patterns	6
80		Common Modeling Elements.....	6
81		Production Rules	7
82	6.1	Key Concepts of the Specification Schema	7
83	6.2	How to use the ebXML Business Process Specification Schema.....	10
84	6.3	How Specification Schema is used with other ebXML specifications	11
85	6.4	How to design collaborations and transactions, re-using at design time	13
86	6.4.1	Specify a Business Transaction and its Document Flow	14
87	6.4.2	Specify a Binary Collaboration.....	20
88	6.4.3	Specify a MultiParty Collaboration.....	23
89	6.4.4	Specify a Choreography	25
90	6.4.5	The whole model.....	27
91	6.5	Core Business Transaction Semantics.....	29
92	6.5.1	Defining and using transaction and interaction patterns	30
93	6.5.2	Transaction Patterns	30
94	6.5.3	Parameters required for CPP/CPA	30
95	6.5.4	Delivery Channel selection parameters:.....	31
96	6.5.5	Reliability	32
97	6.5.6	Synchronous or Asynchronous.....	32
98	6.5.7	BSI service level parameters:.....	33
99	6.6	Run time transaction semantics	34
100	6.6.1	Timeouts.....	35
101	6.6.2	Exceptions	36
102	6.6.3	ControlException	36
103	6.6.4	Business Protocol Exceptions (a.k.a. ProcessException).....	37
104	6.7	Runtime Collaboration Semantics.....	38
105	6.8	Where the ebXML Specification Schema May Be Implemented.....	38

106	7	UML Element Specification.....	38
107	7.1	Business Collaborations	38
108	7.1.1	MultiPartyCollaboration.....	39
109	7.1.2	BusinessPartnerRole.....	39
110	7.1.3	Performs	39
111	7.1.4	AuthorizedRole	40
112	7.1.5	BinaryCollaboration	41
113	7.1.6	BusinessActivity.....	42
114	7.1.7	BusinessTransactionActivity.....	42
115	7.1.8	CollaborationActivity.....	43
116	7.2	Business Transactions	43
117	7.2.1	BusinessTransaction.....	43
118	7.2.2	Business Action.....	45
119	7.2.3	RequestingBusinessActivity.....	46
120	7.2.4	RespondingBusinessActivity	46
121	7.3	Document Flow	47
122	7.3.1	Document Flow	47
123	7.3.2	DocumentType	48
124	7.3.3	Schema	48
125	7.3.4	Attachment	49
126	7.4	Choreography within Collaborations.	49
127	7.4.1	BusinessState.....	49
128	7.4.2	Transition	50
129	7.4.3	Start	50
130	7.4.4	TerminalState	51
131	7.4.5	Success	51
132	7.4.6	Failure.....	51
133	7.4.7	Fork	52
134	7.4.8	Join	52
135	7.4.9	Guard.....	52
136	7.5	Definition and Scope.....	53
137	7.6	Collaboration Specification Rules.....	53
138	7.6.1	Well-formedness Rules	53
139	8	Specification Schema – (DTD)	55
140	8.1	DTD.....	55
141	8.2	Documentation for the DTD	59
142	8.3	XML to UML cross-reference.....	82
143	8.4	Scoped Name Reference	84
144	8.5	Sample XML document against above DTD	85
145	9	Common Modeling Elements.....	91
146	9.1	Data typing	91
147	9.1.1	Global Data types	91
148	9.1.2	Local Datatypes.....	94
149	9.2	Business signal structures.....	94
150	9.2.1	ReceiptAcknowledgment DTD.....	94

151	9.2.2	AcceptanceAcknowledgement DTD.....	94
152	9.2.3	Exception Signal DTD	95
153	10	Production Rules	96
154	11	References	98
155	12	Disclaimer	98
156	13	Contact Information	99
157		Copyright Statement.....	100
158			

159 **4 Introduction**

160 **Executive Summary**

161

162 The ebXML Specification Schema provides a standard framework by which business
163 systems may be configured to support execution of business collaborations consisting of
164 business transactions. It is based upon prior UN/CEFACT work, specifically the
165 metamodel behind the UN/CEFACT Unified Modeling Methodology (UMM) defined in the
166 N90 specification.

167 The Specification Schema supports the specification of Business Transactions and the
168 choreography of Business Transactions into Business Collaborations. Each Business
169 Transaction can be implemented using one of many available standard patterns. These
170 patterns determine the actual exchange of messages and business signals between the
171 partners to achieve the required electronic commerce transaction.

172 The current version of the specification schema addresses collaborations between two
173 parties (Binary Collaborations).

174 It is anticipated that a subsequent version will address additional features such as the
175 semantics of economic exchanges and contracts, more complex multi-party
176 choreography, and context based content.

177 **4.1 Summary of Contents of Document**

178 This document describes the ebXML Specification Schema

179 This document describes the Specification Schema, both in its UML form and in
180 its DTD form.

181 The document first introduces general concepts and semantics, then applies
182 these semantics in a detail discussion of each part of the model. The document
183 then specifies all elements in the UML form, and then in the XML form.

184 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD,
185 SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in
186 this document, are to be interpreted as described in RFC 2119 [Bra97].

187

188 **4.2 Audience**

189 The primary audience is business process analysts. We define a business
190 process analyst as someone who interviews business people and as a result
191 documents business processes in unambiguous syntax.

192 An additional audience is designers of business process definition tools who
193 need to specify the conversion of user input in the tool into the XML
194 representation of the Specification Schema.

195 The audience is not business application developers.

196 **4.3 Related Documents**

197 As mentioned above, other documents provide detailed definitions of some of the
198 components of the ebXML Specification Schema and of their inter-relationship.
199 They include ebXML Specifications on the following topics:

200

- 201 • ebXML Technical Architecture, version 1.0
- 202 • ebXML Core Components, version 1.0
- 203 • ebXML Collaboration Protocol Profile and Agreement, version 1.0
- 204 • ebXML Business Process Analysis Worksheets

205 **4.4 Prerequisites**

206 It is assumed that the audience will be familiar with or have knowledge of the
207 following technologies and techniques:

- 208 • Business process modeling techniques and principles
- 209 • The UML syntax and semantics
- 210 • The Extensible Markup Language (XML)

211 5 Design Objectives

212 5.1 Goals/Objectives/Requirements/Problem Description

213 Business process models specify interoperable business processes that allow
214 business partners to collaborate. Business process models for e-business must
215 be turned into software components that collaborate on behalf of the business
216 partners.

217 The goal of the ebXML Specification Schema is to provide the bridge between e-
218 business process modeling and specification of e-business software
219 components.

220 The ebXML Specification Schema provides for the nominal set of specification
221 elements necessary to specify a collaboration between business partners, and to
222 provide configuration parameters for the partners' runtime systems in order to
223 execute that collaboration between a set of e-business software components.

224 The *Specification Schema* is available in two stand-alone representations, a UML
225 profile, and a DTD. Users of the Specification Schema will create business
226 process specifications as either UML diagrams, or Extensible Markup Language
227 (XML) documents.

228 The objective of the UML based *Specification Schema* is to provide sufficient
229 functionality for business people to model a collaboration. The objective of the
230 XML based *Specification Schema* is to provide sufficient functionality for
231 technical people to unambiguously derive from that collaboration model a set of
232 runtime configuration parameters. The UML and XML based versions of the
233 *Specification Schema* are unambiguously mapped to each other.

234 5.2 Caveats and Assumptions

235 This specification is designed to specify the run time aspects of a business
236 collaboration.

237 It is not intended to incorporate a methodology, and does not directly prescribe
238 the use of a methodology. However, if a methodology is to be used, it is
239 recommended that it be UN/CEFACT Unified Modeling Methodology (UMM).

240 The Specification Schema does not by itself define Business Documents
241 Structures. It is intended to work in conjunction with already existing Business
242 Document definitions, and/or the Business Document model included in the
243 ebXML Core Components specifications.

244

245 5.2.1 Relationship between Specification Schema and UMM

246

247 The UMM Meta Model is a description of business semantics that allows Trading
248 Partners to capture the details for a specific business scenario (a Business
249 Process) using a consistent modeling methodology. A Business Process
250 describes in detail how Trading Partners take on shared roles, relationships and
251 responsibilities to facilitate interaction with other Trading Partners. The

252 interaction between roles takes place as a choreographed set of Business
253 Transactions. Each Business Transaction is expressed as an exchange of
254 electronic Business Documents. The sequence of the exchange is determined
255 by the Business Process, and by messaging and security considerations.
256 Business Documents are composed from re-useable business information
257 components. At a lower level, Business Processes can be composed of re-
258 useable Core Processes, and Business Objects can be composed of re-useable
259 Core Components.

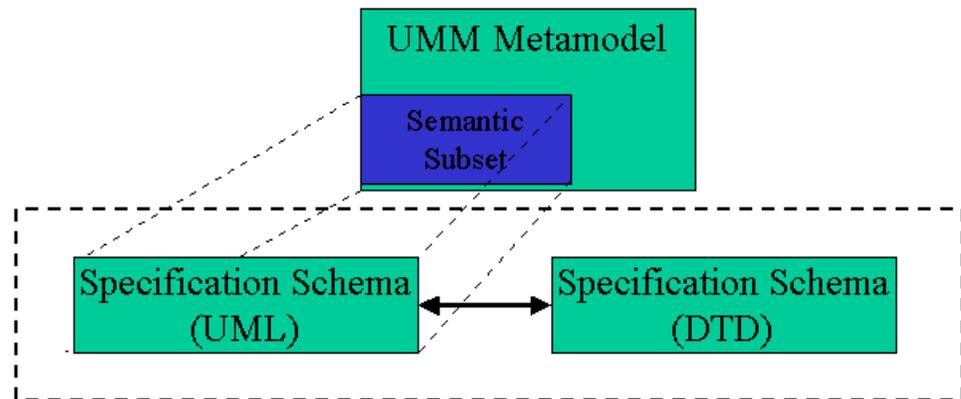
260 The UMM Meta Model supports requirements, analysis and design viewpoints
261 that provide a set of semantics (vocabulary) for each viewpoint and forms the
262 basis of specification of the semantics and artifacts that are required to facilitate
263 business process and information integration and interoperability.

264 An additional view of the metamodel, the Specification Schema, is also provided
265 to support the direct specification of the nominal set of elements necessary to
266 configure a runtime system in order to execute a set of ebXML business
267 transactions. By drawing out modeling elements from several of the other views,
268 the Specification Schema forms a semantic subset of the UMM Meta Model. The
269 Specification Schema is available in two stand-alone representations, a UML
270 profile, and a DTD.

271 The relationship between the UMM Meta Model and the ebXML Specification
272 Schema can be shown as follows:

273

Figure 1. Relationship between Metamodel and Specification Schema



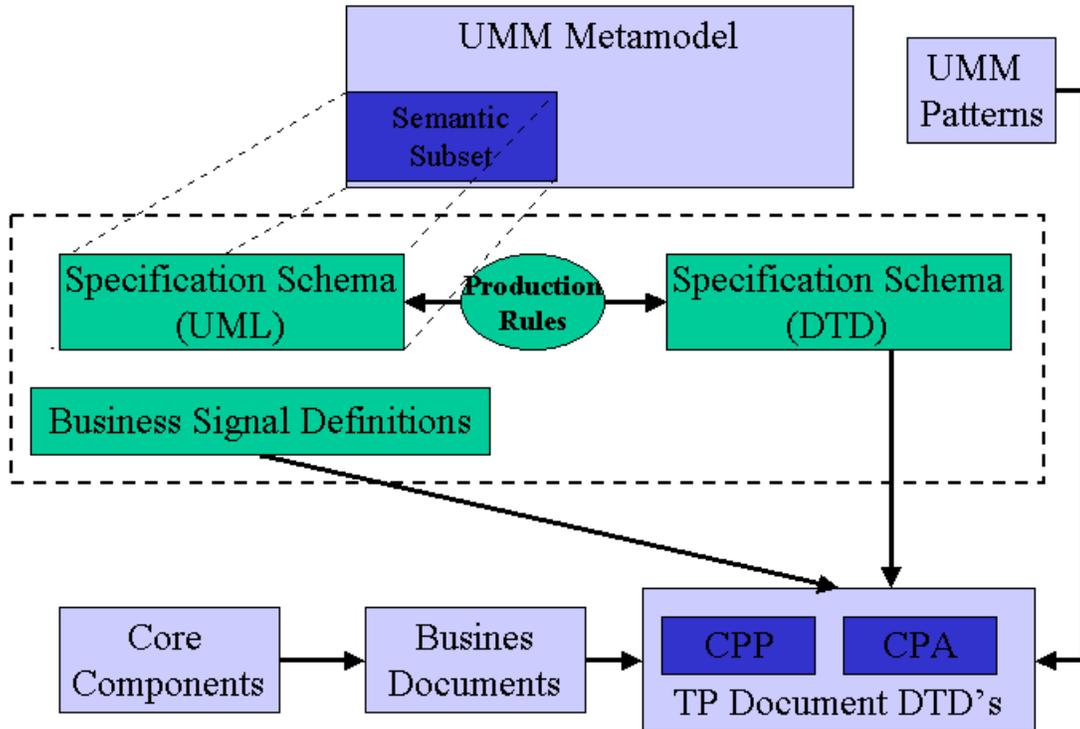
274

275 6 System Overview

276 The *Specification Schema* supports the specification of Business Transactions
 277 and the choreography of Business Transactions into Business Collaborations.
 278 Each Business Transaction can be implemented using one of many available
 279 standard patterns. These patterns determine the actual exchange of messages
 280 and business signals between the partners to achieve the required electronic
 281 commerce transaction.

282 To help specify the patterns *the Specification Schema* is accompanied by a set of
 283 modeling elements common to those standard patterns. The full specification,
 284 thus, of a business process consists of a business process model specified
 285 against the *Specification Schema* and an identification of the desired pattern(s).
 286 This full specification is then the input to the formation of trading partner
 287 Collaboration Protocol Profiles and Collaboration Protocol Agreements.

288 This can be shown as follows:



289

290

Figure 2. Relationship of specification schema to TP and Core Components

291

292 As the figure shows, the architecture of the ebXML Specification Schema
 293 consists of the following functional components (shown inside the dotted box):

294

- UML Specification Schema
- DTD specification Schema
- Business Signal Definitions
- Production Rules needed for the generation of UML specification into a XML Specification Document

295

296

297

298

299 Together these components allow you to fully specify all the run time aspects of a
 300 business process and the accompanying information model.

301

In addition, the UMM provides a set of Business Transaction Interaction Patterns

302

The Specification Schema does not by itself define Business Documents. Rather
 303 it points to already existing Business Document definitions. Such definitions may
 304 have been defined based on ebXML core components, or may be supplied from
 305 some other source.

306

This run time business process and information specification is then incorporated
 307 with the associated Business Document definitions into trading partner

308 Collaboration Protocol Profiles (CPP) and Collaboration Protocol Agreements
309 (CPA). Within these profiles and agreements are then added further technical
310 parameters resulting in a full specification of the run-time software at each
311 trading partner.

312 Each of the components in the Specification Schema is described below:

313 ***UML Specification Schema***

314 The UML Specification Schema is a semantic subset of the metamodel behind
315 UMM as specified in UN/CEFACT TMWG's N90, expressed as a standalone
316 UML profile. The UML Specification Schema will through the application of
317 production rules produce an XML Specification Document is analytically,
318 semantically and functionally equivalent to one arrived at by modeling the same
319 subset through the use of UMM and its associated production rules.

320 ***DTD Specification Schema***

321 The DTD Specification Schema is an isomorphic definition of the UML
322 Specification Schema. The DTD Specification Schema seeks to guarantee that a
323 XML Specification Document is analytically, semantically and functionally
324 equivalent to a UML Specification Model of the same business process.

325 This version of the specification is expressed as a DTD, and some of the
326 constraints may need to be stated separately, in plain text. It is the intent to
327 migrate to W3C schema, as soon as it becomes available as a standard. At that
328 point such constraints, where possible, will be built into the schema.

329 ***Business Process Interaction Patterns***

330 ebXML business service interfaces are configured to execute the business
331 processes defined against the specification schema. They do so by exchanging
332 ebXML messages and business signals. The Business Process Interaction
333 Patterns set forth in Chapter [11] of the UMM N90 document illustrate the
334 permissible permutations of message sequences as determined by the type of
335 business transaction defined and the timing policies specified in the transactions.
336 Those timing policies are expressed by use of the security and timing parameters
337 provided in this document.

338 ***Common Modeling Elements***

339 The Common Modeling Elements specifies the modeling elements, and their
340 interrelationships, that are used to specify and use Interaction Patterns.

341 Business signals are such common modeling elements.

342 Business signals are application level documents that 'signal' the current state of
343 the business transaction. These business signals have specific business purpose
344 and are separate from lower protocol and transport signals.

345 However, business signals are 'constant' and do not vary from transaction to
346 transaction. Thus, they can be defined once and for all as part of the common
347 model elements.

348 ***Production Rules***

349 The Specification Production rules provide the prescriptive definition necessary
350 to translate a UML Specification Model into an XML Specification Document and
351 the well-formed rules necessary to populate an XML Specification Document.

352 Separately, it is expected that a set of production rules will be constructed for the
353 production of an XML Specification Document from a set of UML diagrams
354 constructed through the use of UMM.

355

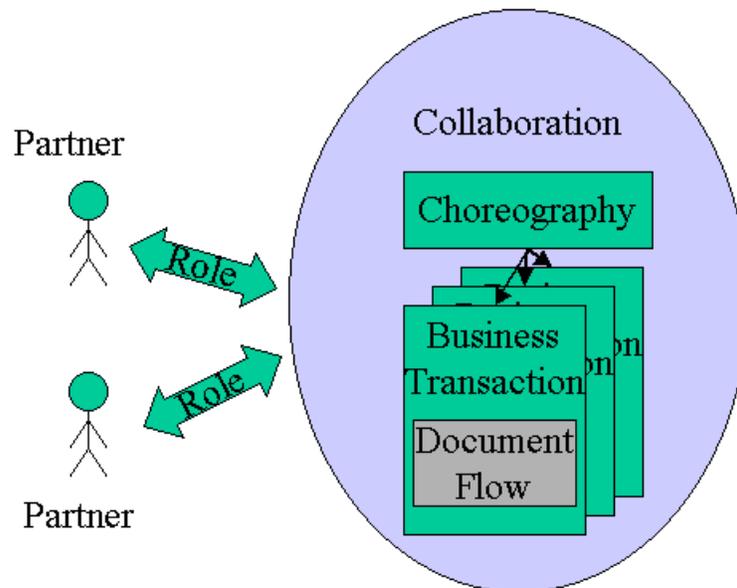
356 ***6.1 Key Concepts of the Specification Schema***

357

358 The ebXML Specification Schema provides the semantics, elements, and
359 properties necessary to define business collaborations.

360 A business collaboration consists of a set of roles collaborating through a set of
361 choreographed transactions by exchanging business documents.

362 These basic semantics of a business collaboration can be shown as follows:



363

364

Figure 3. Basic Semantics of a business collaboration

365

366 Two or more business partners participate in the business collaboration through
 367 roles. The roles interact with each other through Business Transactions. The
 368 business transactions are sequenced relative to each other in a Choreography.
 369 Each business transaction consists of one or two predefined Business Document
 370 Flows.

371 The following section describes the concepts of a Business Collaboration, a
 372 Business Transaction, a Business Document Flow, and a Choreography

373

1. Business Collaborations

374

A business collaboration is a set of Business Transactions between
 375 business partners. Each partner plays one or more roles in the
 376 collaboration.

377

The Specification Schema supports two levels of business collaborations,
 378 Binary Collaborations and Multiparty Collaborations.

379

Binary Collaborations are between two roles only.

380

Multiparty Collaborations are among more than two roles, but such
 381 Multiparty Collaborations are always synthesized from two or more Binary
 382 Collaborations. For instance if Roles A, B, and C collaborate and all

383 parties interact with each other, there will be a separate Binary
384 Collaboration between A and B, one between B and C, and one between
385 A and C. The Multiparty Collaboration will be the synthesis of these three
386 Binary Collaborations.

387 Binary Collaborations are expressed as a set of BusinessActivities
388 between the two roles. Each BusinessActivity reflects a state in the
389 collaboration. The BusinessActivity can be 'atomic', i.e. the activity of
390 conducting an atomic BusinessTransaction, or 'composite', i.e. the activity
391 of conducting another Binary Collaboration. In either case the activities
392 can be choreographed as per below.

393 The ability of a Binary Collaboration to have activities that in effect are
394 executing other Binary Collaborations, is the key to recursive
395 compositions of Binary Collaboration, and to the re-use of Binary
396 Collaborations.

397 In essence each Binary Collaboration is a re-useable protocol between
398 two roles.

399 2. Business Transactions

400 A Business Transaction is the atomic unit of work in a trading
401 arrangement between two business partners. A Business Transaction is
402 conducted between two parties playing opposite roles in the transaction.
403 The roles are always a requesting role and a responding role.

404 Like a Binary Collaboration, a Business Transaction is a re-useable
405 protocol between two roles. The way it is re-used is by referencing it from
406 a Binary Collaboration. In essence the roles of the Binary Collaboration
407 are assigned to the execution of the Business Transaction.

408 Unlike a Binary Collaboration, however, the Business Transaction is
409 atomic, it cannot be decomposed into lower level Business Transactions.

410 A Business Transaction is a very specialized and very constrained
411 protocol, in order to achieve very precise and enforceable transaction
412 semantics. These semantics are expected to be enforced by the software
413 managing the transaction, i.e. a Business Service Interface (BSI).

414 A Business Transaction will always either succeed or fail. If it succeeds it
415 may be legally binding for the two partners. If it fails it is null and void,
416 and each partner must relinquish any mutual claim established by the
417 transaction. This can be thought of as 'rolling back' the transaction upon
418 failure.

419 3. Business Document Flows

420 A business transaction is realized as Business Document Flows between
421 the requesting and responding activities. There is always a requesting
422 Business Document, and optionally a responding Business Document,
423 depending on the desired transaction semantics, e.g. one-way notification
424 vs. two-way conversation.

425 Actual document definition is achieved using the ebXML core component
426 specifications, or by some methodology external to ebXML but resulting in
427 a DTD or Schema that a specification schema instance can point to.

428 4. Choreography

429 The Business Transaction Choreography describes the ordering and
430 transitions between business transactions or sub collaborations within a
431 binary collaboration. In a UML tool this can be done using a UML activity
432 diagram. The choreography is described in the Specification Schema
433 using activity diagram concepts such as start state, completion state,
434 activities, synchronizations, transitions between activities, and guards on
435 the transitions.

436 5. Patterns

437 The specification schema provides a set of unambiguous semantics
438 within which to specify transactions and collaborations. Within these
439 semantics the using community have flexibility to specify very different
440 transactions and collaborations. The use of patterns combines this
441 flexibility with a consistency that facilitates faster design, faster
442 implementation, and enables generic processing.

443 A set of parameters determine the semantics of a business transaction.

444 A set of predefined transaction patterns can be found in UMM.

445 Re-use, recursion, and patterns are among the key concepts of the Specification
446 Schema, the following section will illustrate these key concepts.

447 **6.2 How to use the ebXML Business Process Specification** 448 **Schema**

449 The ebXML Specification Schema should be used wherever software is being
450 specified to Business Collaborations.

451 The ebXML Specification Schema is used to specify the business process
452 related configuration parameters for configuring software to execute these
453 collaborations

454 This section discusses

- 455 • How the Specification Schema fits in with other ebXML specifications.
- 456 • How to use the Specification Schema at design time, either for specifying
457 brand new collaborations and transactions, or for re-using existing ones.
- 458 • How to specify core transaction semantics and parameters needed for a
459 CPP/CPA.
- 460 • Run-time transaction and collaboration semantics that the Specification
461 Schema specifies and the Business Service Interface (BSI) is expected to
462 manage.

463 **6.3 How Specification Schema is used with other ebXML**
464 **specifications**

465

466 The ebXML Specification Schema provides the semantics, elements, and
467 properties necessary to define Business Collaborations.

468

469 A collaboration consists of a set of roles collaborating through a set of
choreographed transactions by exchanging business documents.

470

471 The business documents are defined at the intersection between the business
472 process model and the information model. At ebXML that means that the
473 Business Documents are assembled from lower level information structures
474 known as core components. The assembly is based on a set of contexts, many
475 of which are provided by the business processes, i.e. collaborations that use the
documents in their Document Flows.

476

477 The combination of the business process model and the information model
478 become the basis against which partners can make agreements on conducting
electronic business with each other.

479

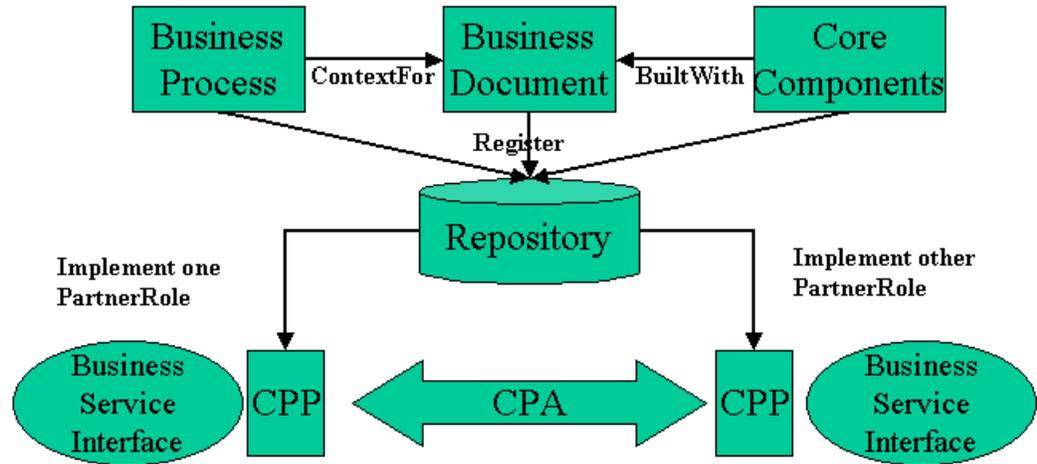
480 Specifically The ebXML Specification Schema is intended to provide the
481 business process specification for the formation of a partner Collaboration
Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA).

482

483 The actual configuration of the Business Service Interface (BSI) is done
according to the ebXML Collaboration Protocol Profile specification.

484

This process can be illustrated as follows:



485

486

Figure 4. Specification Schema and other ebXML Specifications

487

488 The Specification Schema allows the specification of business processes. The
 489 resulting business process specification is stored in the repository, and provides
 490 context for document definition. It then provides the business parameters for
 491 CPP and CPA creation.

492 A set of specification rules have been established to properly constrain the
 493 expression of a business process and information model in a way that can be
 494 directly incorporated into a trading partner Collaboration Protocol Profile and
 495 Agreement.

496 A user would use a UML tool to create a model instance against the Specification
 497 Schema, and would then use the production rules to produce the XML version of
 498 the model, compliant with the DTD version of the specification schema.

499 Alternatively a user would use an XML based tool to produce the XML version
 500 directly. Production rules would then aid in converting into XMI, so that it could be
 501 loaded into a UML tool, if required.

502 In either case, the XML version gets stored in the ebXML repository and
 503 registered in the ebXML registry for future retrieval when implementers want to
 504 establish trading partner Collaboration Protocol Profile and Agreement. At that

505 point the XML document, or the relevant parts of it, are simply imbedded in or
506 referenced by the CPP and CPA XML documents.
507 Guided by the CPP and CPA specifications the resulting XML document then
508 becomes the configuration file for one or more Business Service Interfaces (BSI),
509 i.e. the software that will actually manage either partner's participation in the
510 collaboration.

511 **6.4 How to design collaborations and transactions, re-using at** 512 **design time**

513

514 This section describes the Specification Schema modeling relationships by
515 building a complete Multiparty Collaboration from the bottom up, as follows:

- 516 1. Specify a Business Transaction
- 517 2. Specify Document Flows for a Business Transaction
- 518 3. Specify a Binary Collaboration re-using the Business Transaction
- 519 4. Specify a Choreography for the Binary Collaboration
- 520 5. Specify a higher level Binary Collaboration re-using the lower level Binary
521 Collaboration
- 522 6. Specify a Multiparty Collaboration re-using Binary Collaborations

523 Although this section, for purposes of introduction, discusses the model from the
524 bottom up, the Specification Schema very much is intended for modeling from
525 the top down, re-using existing lower level content as much as possible.

526

540 Activity. These roles become explicit when the transaction is used within
541 a Business Transaction Activity within a Binary Collaboration.
542 There is always a Request Document Flow.
543 Whether a Response Document Flow is required is part of the definition
544 of the Business Transaction. Some Business Transactions need this type
545 of request and response, typically for the formation of a contract or
546 agreement. Other Business Transactions are more like notifications, and
547 have only a Request Document Flow.
548 An abstract superclass, Business Action, is the holder of attributes that
549 are common to both Requesting Business Activity and Responding
550 Business Activity.

551 6.4.1.2 Sample syntax

552 Here is a simple notification transaction with just one document flow:

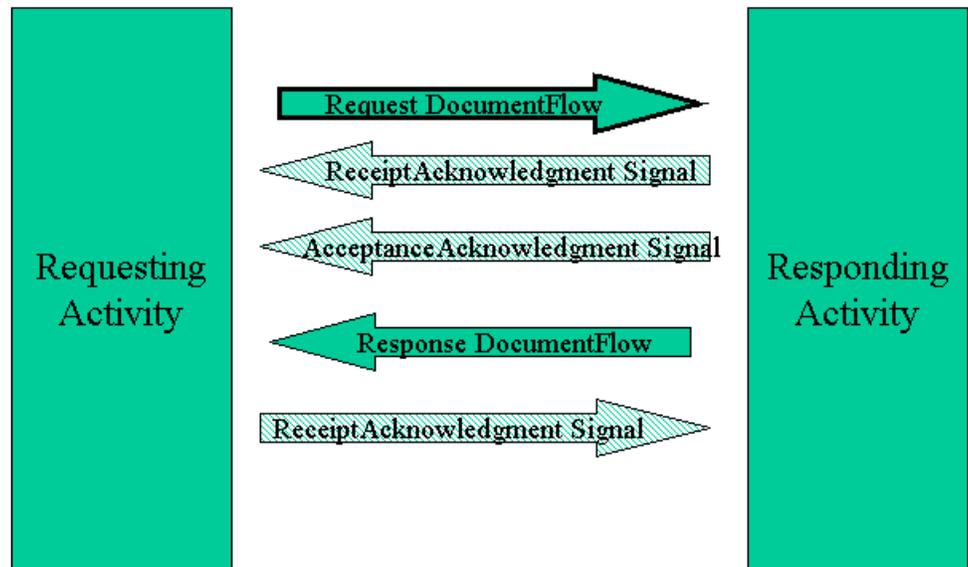
```
553 <BusinessTransaction name="Notify of advanceshipment">  
554     <RequestingBusinessActivity name="">  
555         <DocumentFlow isSuccess="true"  
556             documentType name="ASN" />  
557     </RequestingBusinessActivity>  
558     <RespondingBusinessActivity name="">  
559     </RespondingBusinessActivity>  
560 </BusinessTransaction>  
561
```

562 Associated with each Document Flow can be one or more business
563 signals acknowledging the Document Flow. These acknowledgment
564 signals are not modeled explicitly but parameters associated with the
565 transaction specify whether the signals are required or not.

566 The possible Document Flows and business signals within a Business
567 Transaction can be shown as follows:

568

569



570

571 **Figure 6. Possible document flows and signals and their sequence**

572

573

574

These acknowledgment signals (a.k.a. Business Signals) are application level documents that 'signal' the current state of the business transaction.

575

576

577

578

579

Whether a receiptAcknowledgement and/or acceptanceAcknowledgement signal are sent is part of the selection of an interaction pattern for the Business Transaction. These business signals have specific business purpose and are separate from lower protocol and transport signals.

580

581

582

583

584

585

Receipt acknowledgement business signal. The UN/EDIFACT model Trading Partner Agreement (TPA) suggests that a partners should agree on the point at which a message can be "said" to be properly received and this point should be when a receiving partner can "read" a message¹. The property *isIntelligibleCheckRequired* allows partners to agree that a message should be "readable" before its receipt is verified².

586

587

Acceptance Acknowledgement business signal. The UN/EDIFACT model TPA suggests that partners should agree on the point at which a

¹ The UN/ECE defines this to be the point after which a message passes a structure/ schema validity check. This is not a necessary condition for verifying proper receipt, only accessibility is.

588 message can be "said" to be accepted for business processing and this
589 point should be after the contents of a business document have passed a
590 business rule validity check.³

591 6.4.1.3 Sample syntax

592 Here is a slightly more complex transaction with two Document Flows and
593 three business signals.

594 The request requires both receipt and acceptance acknowledgement, the
595 response requires only receipt acknowledgement. "P2D" is a W3C/ISO
596 standard and means Period=2 Days. P3D means Period=3 Days, P5D
597 means Period=5 Days. These periods are all measured from original
598 sending of request.

```
599 <BusinessTransaction name="Create Order">  
600     <RequestingBusinessActivity name=" "  
601         isNonRepudiationRequired="true"  
602         timeToAcknowledgeReceipt="P2D"  
603         timeToAcknowledgeAcceptance="P3D">  
604         <DocumentFlow isSuccess="true"  
605             documentType="Purchase Order"/>  
606     </RequestingBusinessActivity>  
607     <RespondingBusinessActivity name=" "  
608         isNonRepudiationRequired="true"  
609         timeToAcknowledgeReceipt="P5D">  
610         <DocumentFlow isSuccess="true"  
611             documentType="PO Acknowledgement"/>  
612         </DocumentFlow>  
613     </RespondingBusinessActivity>  
614 </BusinessTransaction>  
615
```

616

617 6.4.1.4 Specifying Document Flows

618

619 Request Document Flows and Response Document Flows contain
620 Business Documents that pertain to the Business Transaction. The model
621 for this is shown below. Business Documents all have unique structures.
622 Business signals, however always have the same structure, defined once
623 and for all as part of the common modeling elements.

624

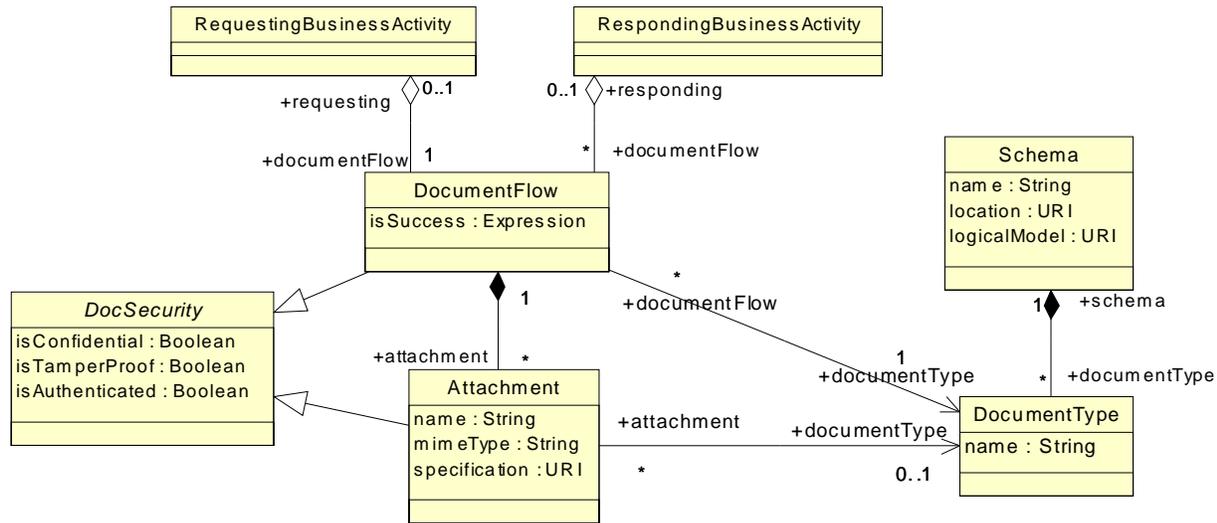
625

626

627

³ This is the convention specified for RosettaNet and UN/CEFACT N90 commercial transactions.

628



629

630

Figure 7. UML semantics of a Document Flow

631

632

633

634

635

636

637

638

639

Document Flows are named. There is always only one named Document Flow for a Requesting Activity. There may be zero, one, or many named Document Flows for a Responding Activity. For example, the Response Document Flows for a purchase order transaction might be named PurchaseOrderAcceptance, PurchaseOrderDenial, and PartialPurchaseOrderAcceptance. In the actual execution of the purchase order transaction, however, only one of the defined possible responses will be sent.

640

641

642

The Document Flow represents the flow of documents between the activities. Each Document Flow carries exactly one primary Business Document. The document is specified in terms of its DocumentType.

643

644

A Document Type is defined in a Schema. This may be an ebXML Schema, or a Schema supplied by an outside source.

645

646

647

648

A Document Flow can optionally have one or more attachments, all related to the primary Business Document. The document and its attachments in essence form one transaction in the payload in the ebXML TRP message.

649 6.4.1.5 Sample syntax

650

651

652

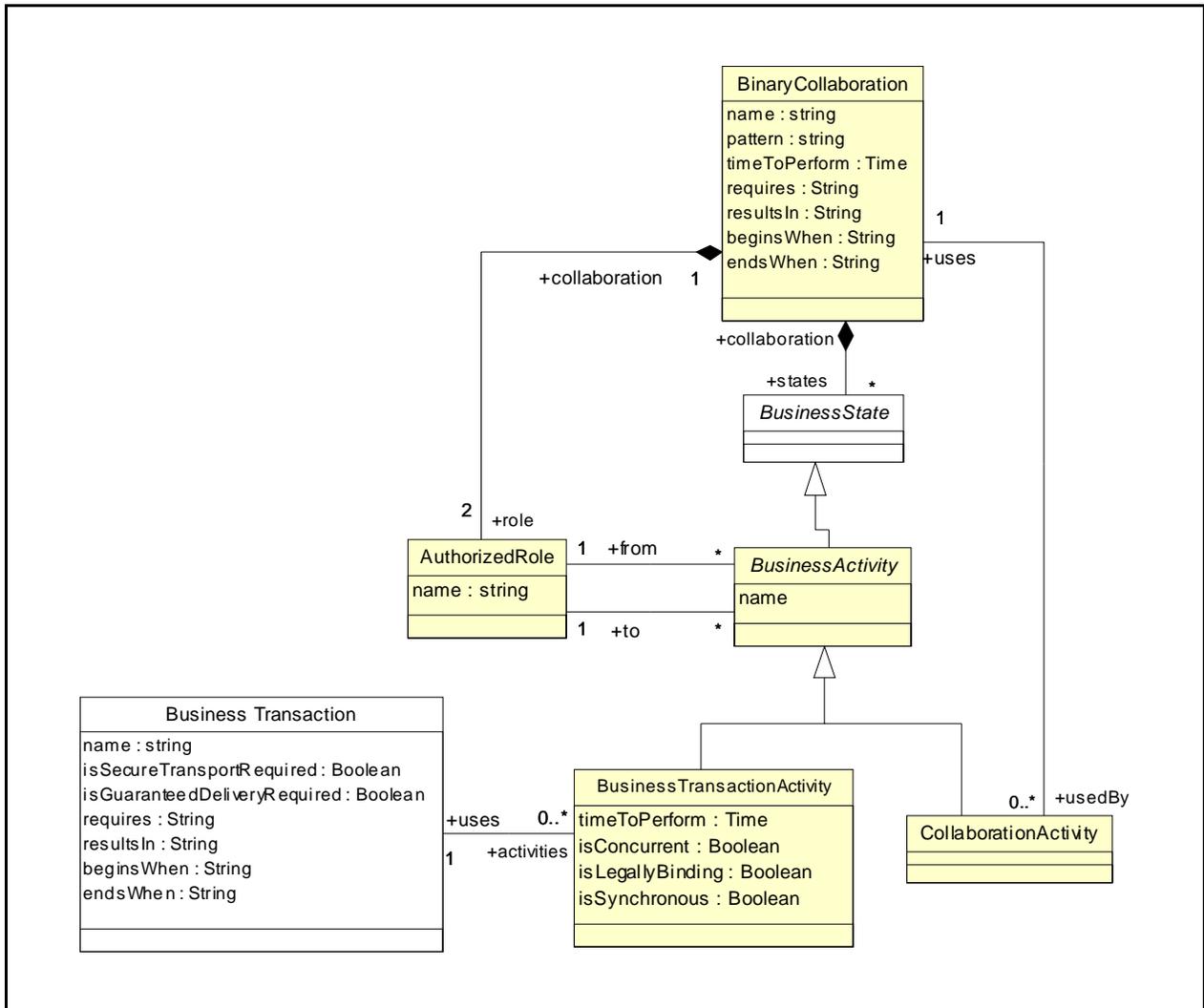
653

654

This example shows a business transaction with one request and two possible responses, a success and a failure. The request has an attachment. All the documents are defined in a named schema, and the document types are fully qualified with the schema name.

```
655     <Schema name="ebXML1.0" location="someplace"
656           logicalModel="someplaceAlso">
657       <DocumentType name=" Purchase Order "/>
658       <DocumentType name=" PO Acknowledgement "/>
659       <DocumentType name=" PO Rejection "/>
660       <DocumentType name="Delivery Instructions"/>
661     </Schema>
662
663     <BusinessTransaction name="Create Order">
664       <RequestingBusinessActivity name=""
665         <DocumentFlow isSuccess="true"
666           documentType="ebXML1.0/PO Acknowledgement">
667         <Attachment
668           name="DeliveryNotes"
669           mimeType="XML"
670           documentType=
671             "ebXML1.0/Delivery Instructions"
672           specification=""
673           isConfidential="true"
674           isTamperProof="true"
675           isAuthenticated="true">
676         </Attachment>
677       </DocumentFlow>
678     </RequestingBusinessActivity>
679     <RespondingBusinessActivity name=""
680       <DocumentFlow isSuccess="true"
681         documentType="ebXML1.0/PO Acknowledgement"/>
682     </DocumentFlow>
683     <DocumentFlow isSuccess="false"
684       documentType=" ebXML1.0/PO Rejection"/>
685     </DocumentFlow>
686   </RespondingBusinessActivity>
687 </BusinessTransaction>
688
```

689 6.4.2 Specify a Binary Collaboration



690

691 **Figure 8. UML semantics of a Binary Collaboration**

692

693

694 6.4.2.1 **Key Semantics of a Binary Collaboration**

695 A Binary Collaboration is always between two roles. These two roles are
 696 called Authorized Roles, because they represent the actors that are
 697 authorized to participate in the collaboration.

698 A Binary Collaboration consists of one or more Business Activities. These
 699 Business Activities are always conducted **between** the two Authorized
 700 Roles of the Binary Collaboration. For each activity one of two roles is
 701 assigned to be the initiator (from) and the other to be the responder (to).

702 A Business Activity can be either a Business Transaction Activity or a
703 Collaboration Activity.

704 A Business Transaction Activity is the performance of a Business
705 Transaction. Business Transactions are re-useable relative to Business
706 Transaction Activity. The same Business Transaction can be performed
707 by multiple Business Transaction Activities in different Binary
708 Collaborations, or even by multiple Business Transaction Activities in the
709 same Binary Collaboration.

710 A Collaboration Activity is the performance of a Binary Collaboration
711 within another Binary Collaboration. Binary Collaborations are re-useable
712 relative to Collaboration Activity. The same Binary Collaboration can be
713 performed by multiple Collaboration Activities in different Binary
714 Collaborations, or even by multiple Collaboration Activities in the same
715 Binary Collaboration.

716 When performing a Binary Collaboration within a Binary Collaboration
717 there is an implicit relationship between the roles at the two levels.
718 Assume that Binary Collaboration X is performing Binary Collaboration Y
719 through Collaboration Activity Q. Binary Collaboration X has Authorized
720 roles Customer and Vendor. In Collaboration Activity Q we assign
721 Customer to be the initiator, and Vendor to be the responder. Binary
722 Collaboration X has Authorized roles Buyer and Seller and a Business
723 Transaction Activity where Buyer is the initiator and Seller the responder.
724 We have now established a role relationship between the roles Customer
725 and Buyer because they are both initiators in activities in the related
726 performing and performed Binary Collaborations.

727 Since a Business Transaction is atomic in nature, the performing of a
728 single Business Transaction through a Business Transaction Activity is
729 also atomic in nature. If the desired semantic is not atomic, then the task
730 should be split over multiple transactions. For instance if it is desired to
731 model several partial acceptances of a request, then the request should
732 be modeled as one transaction within a binary collaboration and the
733 partial acceptance(s) as separate transactions.

734 The CPA/CPP Specification requires that parties agree upon a
735 Collaboration Protocol Agreement (CPA) in order to transact business. A
736 CPA associates itself with a specific Binary Collaboration. Thus, all
737 Business Transactions performed between two parties must be
738 referenced through Business Transaction Activities contained within a
739 Binary Collaboration.

740 Trading partners may wish to indicate that a Business Transaction
741 performed as part of an ebXML arrangement is, or is not, intended to be
742 binding. Under the developing laws of electronic signatures, a declaration
743 of intent to be bound is a key element in establishing the legal
744 equivalence of an electronic message to a signed physical writing.
745 However, in ebXML, the presence or absence of an electronic signature
746 cannot indicate legally binding assent, because it is reserved for use as
747 an assurance of sender identity and message integrity.

748
749 Parties may wish to conduct nonbinding transactions for a variety of
750 reasons, including testing, and the exchange of proposed offers and
751 counteroffers on a non-committal basis so as to discover a possible
752 agreed set of terms. When using tangible signed documents, parties
753 often do so by withholding a manual signature, or using a "DRAFT"
754 stamp. In ebXML, trading partners may indicate that result by use of the
755 "isLegallyBinding" parameter.

756
757 isLegallyBinding is a parameter at the BusinessTransactionActivity level,
758 which means that the performing of a BusinessTransaction within a
759 Binary Collaboration is either legally binding or not.

760
761 As in EDI, the ebXML standard assumes that Business Transactions are
762 intended by the trading parties to be binding unless otherwise indicated.
763 When operating under this standard, parties form binding agreements by
764 exchanging binding messages that agree to terms (e.g., offer and
765 acceptance).

766
767 The "isLegallyBinding" parameter is Boolean, and its default value is
768 "true." Under this standard, the exclusive manner for indicating that a
769 Business Activity is not intended to be binding is to include a "false"
770 value for the "isLegallyBinding" parameter for the transaction activity.
771 That value indicates to the trading parties that the activities conducted
772 in that transaction are only intended as a proposal or test, and that they
773 may not rely on or seek to enforce the other's activity."
774

775 6.4.2.2 Sample syntax

776
777 Here is a simple Binary Collaboration using one of the Business
778 Transactions defined above:

```
779  
780 <BinaryCollaboration name="Firm Order"  
781 timeToPerform="P2D">  
782   <Documentation>  
783     timeToPerform =  
784     Period: 2 days from start of transaction  
785   </Documentation>  
786   <AuthorizedRole name="buyer"/>  
787   <AuthorizedRole name="seller"/>  
788   <BusinessTransactionActivity name="Create Order"  
789     businessTransaction="Create Order"  
790     fromAuthorizedRole="buyer"  
791     toAuthorizedRole="seller"/>  
792 </BinaryCollaboration>  
793
```

794 Here is a slightly more complex Binary Collaboration re-using the same
 795 Business Transaction as the previous Binary Collaboration, and adding the
 796 use of another of the Business Transactions defined above.:

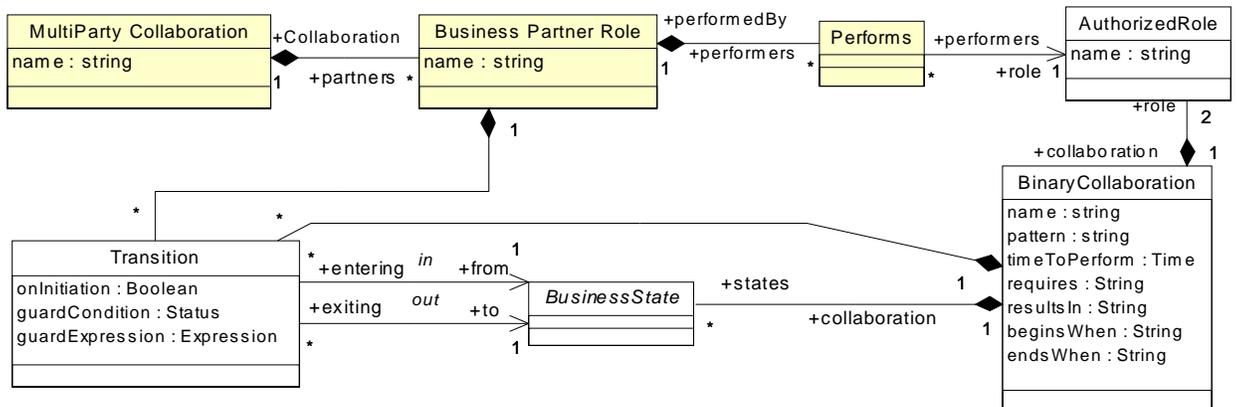
```

797
798 <BinaryCollaboration name="Product Fulfillment"
799   timeToPerform="P5D">
800   <Documentation>
801     timeToPerform =
802     Period: 5 days from start of transaction
803   </Documentation>
804   <AuthorizedRole name="buyer"/>
805   <AuthorizedRole name="seller"/>
806   <BusinessTransactionActivity name="Create Order"
807     businessTransaction="Create Order"
808     fromAuthorizedRole="buyer"
809     toAuthorizedRole="seller"
810     isLegallyBinding="true" />
811   <BusinessTransactionActivity
812     name="Notify shipment"
813     businessTransaction="Notify of advance
814     shipment"
815     fromAuthorizedRole="buyer"
816     toAuthorizedRole="seller" />
817 </BinaryCollaboration>
818
819
820
  
```

820

821 6.4.3 Specify a MultiParty Collaboration

822



823

824

825 **Figure 9. UML semantics of a MultiParty Collaboration**

826

827

828 **6.4.3.1 Key Semantics of a Multiparty Collaboration**

829 A Multiparty Collaboration is a synthesis of Binary Collaborations.

830 A Multiparty Collaboration consists of a number of Business Partner
831 Roles.832 Each Business Partner Role performs one Authorized Role in one of the
833 binary collaborations, or perhaps one Authorized Role in each of several
834 binary collaborations. This is modeled by use of the Performs element.835 This 'Performs' linkage between a Business Partner Role and an
836 Authorized Role is the synthesis of Binary Collaborations into Multiparty
837 Collaborations. Implicitly the Multiparty Collaboration consists of all the
838 Binary Collaborations in which its Business Partner Roles play Authorized
839 Roles.840 Each binary pair of trading partners will be subject to one or more distinct
841 CPAs.842 Within a Multiparty Collaboration, you may choreograph transitions
843 between Business Transactions Activities in different Binary
844 Collaborations, as described below.

845

846 **6.4.3.2 Sample syntax**847 Here is a simple Multiparty Collaboration using the Binary Collaborations
848 defined above.

```

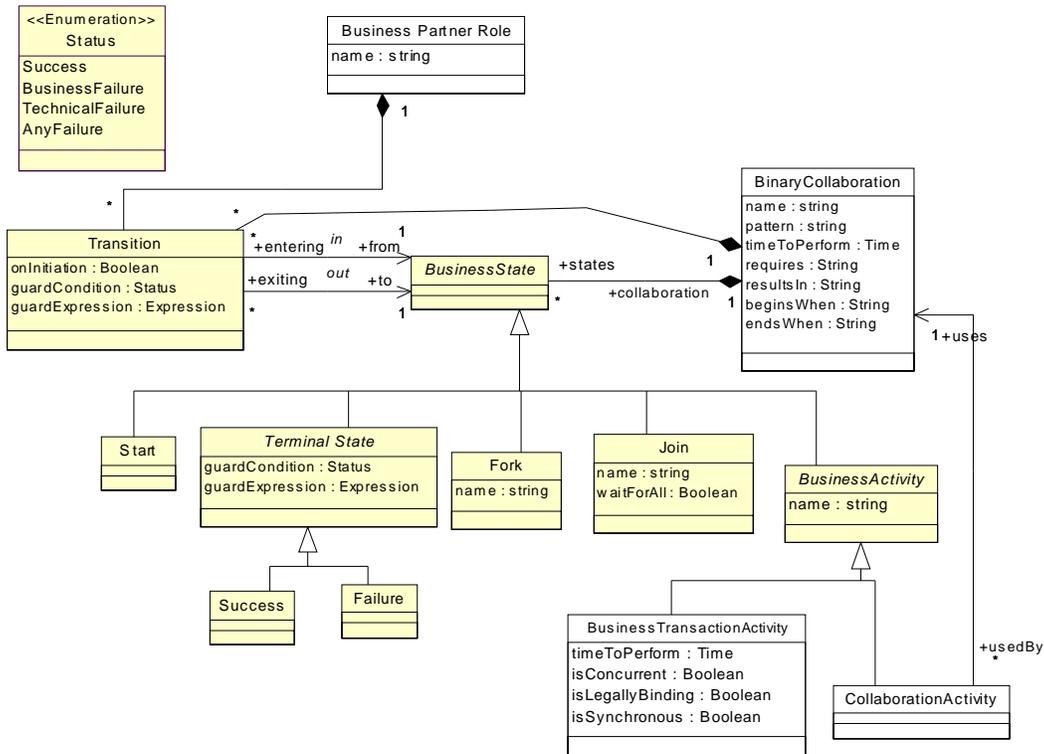
849 <MultiPartyCollaboration name="DropShip">
850   <BusinessPartnerRole name="Customer">
851     <Performs
852       binaryCollaboration="Firm Order"
853       authorizedRole="buyer"/>
854   </BusinessPartnerRole>
855   <BusinessPartnerRole name="Retailer">
856     <Performs
857       binaryCollaboration="Firm Order"
858       authorizedRole="seller"/>
859     <Performs
860       binaryCollaboration="Product Fulfillment"
861       authorizedRole="buyer"/>
862   </BusinessPartnerRole>
863   <BusinessPartnerRole name="DropShip Vendor">
864     <Performs
865       binaryCollaboration="Product Fulfillment"
866       authorizedRole="seller"/>
867   </BusinessPartnerRole>
868 </MultiPartyCollaboration>

```

869

870

871 6.4.4 Specify a Choreography
872



873

874

Figure 10. UML semantics of a Choreography

875

876 6.4.4.1 Key Semantics of a Choreography

877

878

879

A Choreography is an ordering and sequencing of Business Activities within a Binary Collaboration.

880

881

The choreography is specified in terms of Business States, and transitions between those Business States.

882

883

884

885

886

887

A Business Activity is an abstract kind of Business State. Its two subtypes Business Transaction Activity and Collaboration Activity are concrete Business States. The purpose of a Choreography is to order and sequence Business Transaction Activity and/or Collaboration Activity within a Binary Collaboration, or across Binary Collaborations within a Multiparty Collaboration.

888

889

890

There are a number of auxiliary kinds of Business States that facilitate the choreographing of Business Activities. These include a Start state, a Completion state (which comes in a Success and Failure flavor), a Fork

891 state and a Synchronization state. These are all equivalent to
 892 diagramming artifacts on a UML activity chart.

893 Transitions are between Business States. Transitions can be gated by
 894 Guards. Guards can refer to the status of the Document Flow that caused
 895 the transition, the type of Document sent, the content of the document, or
 896 postconditions on the prior state.

897 A Transition can also be used to create nested
 898 BusinessTransactionActivities. A nested BusinessTransactionActivity is
 899 one where a first transition happens after the receipt of the request in the
 900 first transaction, and then the entire second transaction is performed
 901 before returning to the first transaction to send the response back to the
 902 original requestor. The flag 'onInitiation' in the
 903 BusinessTransactionActivity is used for this purpose. Nested
 904 BusinessTransactionActivity are typically within a multiparty collaboration.
 905 In essence an Authorized Role in one Binary Collaboration receives a
 906 request, then turns around and becomes the requestor in an other Binary
 907 Collaboration before coming back and sending the response in the first
 908 Binary Collaboration.

909 Concurrency is a parameter that governs the flow of transactions. Unlike
 910 the security and timing parameters it does not govern the internal flow of
 911 a transaction, rather it determines whether multiple instances of that
 912 transaction type can be 'open' at the same time as part of the same
 913 business transaction activity. IsConcurrent the parameter that governs
 914 this. It is at business transaction activity level.

915

916 6.4.4.2 Sample syntax

917

918 Here is the same Binary Collaboration as used before, with choreography
 919 added at the end. There is a transition between the two, a start and two
 920 possible outcomes of this collaboration, success and failure:

```

921 <BinaryCollaboration name="Product Fulfillment"
922   timeToPerform="P5D">
923   <Documentation>
924     timeToPerform =
925     Period: 5 days from start of transaction
926   </Documentation>
927   <AuthorizedRole name="buyer"/>
928   <AuthorizedRole name="seller"/>
929   <BusinessTransactionActivity name="Create Order"
930     businessTransaction="Create Order"
931     fromAuthorizedRole="buyer"
932     toAuthorizedRole="seller"/>
933   <BusinessTransactionActivity
934     name="Notify shipment"
935     businessTransaction="Notify of advance
936     shipment"
  
```

```

937         fromAuthorizedRole="buyer"
938         toAuthorizedRole="seller"/>
939     <Start toBusinessState="Create Order"/>
940     <Transition
941         fromBusinessState="Create Order"
942         toBusinessState="Notify shipment"/>
943     <Success fromBusinessState="Notify shipment"
944         guardCondition="Success"/>
945     <Failure fromBusinessState="Notify shipment"
946         guardCondition="BusinessFailure"/>
947 </BinaryCollaboration>
948

```

949 Here is the same Multiparty Collaboration as defined before, but with a simple
950 choreography (transition) across two Binary Collaborations.

```

951     <MultiPartyCollaboration name="DropShip">
952         <BusinessPartnerRole name="Customer">
953             <Performs
954                 binaryCollaboration="Firm Order"
955                 authorizedRole="buyer"/>
956         </BusinessPartnerRole>
957         <BusinessPartnerRole name="Retailer">
958             <Performs
959                 binaryCollaboration="Firm Order"
960                 authorizedRole="seller"/>
961             <Performs
962                 binaryCollaboration="Product Fulfillment"
963                 authorizedRole="buyer"/>
964             <Transition
965                 fromBinaryCollaboration="Firm Order"
966                 fromBusinessState="Create Order"
967                 topBinaryCollaboration="Product Fulfillment"
968                 toBusinessState="Create Order"/>
969         </BusinessPartnerRole>
970         <BusinessPartnerRole name="DropShip Vendor">
971             <Performs
972                 binaryCollaboration="Product Fulfillment"
973                 authorizedRole="seller"/>
974         </BusinessPartnerRole>
975     </MultiPartyCollaboration>
976

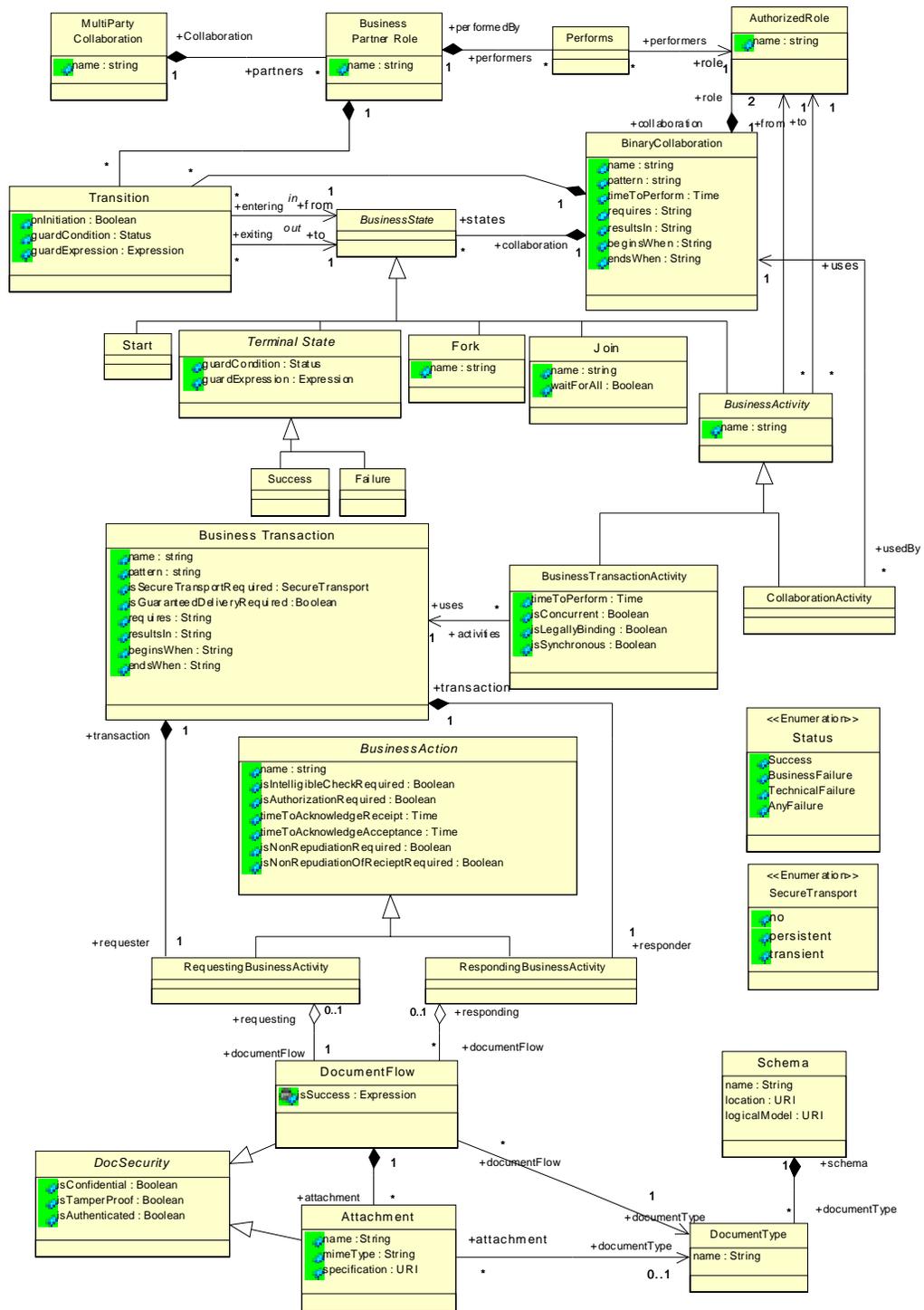
```

977 6.4.5 The whole model

978
979 The following picture shows the above semantics collectively as a UML class
980 diagram. This diagram contains the whole UML version of the Specification
981 Schema

982

Figure 11. Overall Specification Schema as UML class diagram



983

984

985

986 **6.5 Core Business Transaction Semantics**

987 The ebXML concept of a business transaction and the semantics behind it are
988 central to predictable, enforceable commerce. It is expected that any Business
989 Service Interface (BSI) will be capable of managing a transaction according to
990 these semantics.

991 In the ebXML model the business transaction always has the following
992 semantics.

- 993 1. The Business Transaction is a unit of work. All of the interactions in a
994 business transaction must succeed or the transaction must be rolled back
995 to a defined state before the transaction was initiated.
- 996 2. A business transaction is conducted between two business partners
997 playing opposite roles in the transaction. These roles are always the
998 Requesting Role and the Responding Role.
- 999 3. A Business Transaction definition specifies exactly when the Requesting
1000 Activity is in control, when the Responding Activity is in control, and when
1001 control transitions from one to the other. In all Business Transactions
1002 control starts at the Requesting Activity, then transitions to the
1003 Responding Activity, and then returns to the Requesting Activity.
- 1004 4. A business transaction always starts with a request sent out by the
1005 requesting activity.
- 1006 5. The request serves to transition control to the responding role.
- 1007 6. After the receipt of the Request Document Flow, the responding activity
1008 may send a receiptAcknowledgement signal and/or an
1009 acceptanceAcknowledgement signal to the requesting role.
- 1010 7. The responding role then enters a responding activity. During or upon
1011 completion of the responding activity zero or one response is sent.
- 1012 8. The response (if any) transitions control back to the requesting role. If no
1013 response is sent then control transitions back to the requesting role based
1014 on the receipt of a business signal. Regardless which combination of
1015 receiptAcknowledgement and/or acceptanceAcknowledgement is chosen
1016 and/or Response Document Flow is chosen, they always flow in the
1017 sequence just listed, and the last one always transfers control back to the
1018 requesting activity.
- 1019 9. All business transactions succeed or fail. Success or failure depends on:
 - 1020 a. The receipt or non-receipt of the response or business signals
 - 1021 b. The occurrence of time-outs
 - 1022 c. The occurrence of a business exception
 - 1023 d. The occurrence of a control exception

1024 10. Upon receipt of the Response Document Flow the requesting activity may
1025 send a receiptAcknowledgement signal back to the responding role. This
1026 is merely a signal and does not pass control back to the responding
1027 activity.

1028

1029 6.5.1 Defining and using transaction and interaction patterns

1030

1031 Even though all business transactions are governed by very precise semantics,
1032 business transactions can be carried out in many distinctly and succinctly
1033 Specified patterns and with different security and exchange characteristics.

1034 The groups of specification elements that determine the exact pattern of a
1035 business transaction are:

- 1036 • Transaction Patterns and Time-out parameters
- 1037 • Security parameters.
- 1038 • Reliability parameters
- 1039 • Synchronous or asynchronous

1040 6.5.2 Transaction Patterns

1041

1042 Transaction patterns are based on whether a response is required and whether
1043 receiptAcknowledgement and/or acceptanceAcknowledgement are required.

1044 The way to specify that a receiptAcknowledgement is required is to set the
1045 parameter timeToAcknowledgeReceipt to any value other than blank.

1046 The way to specify that a acceptanceAcknowledgement is required is to set the
1047 parameter timeToAcknowledgeAcceptance to any value other than blank.

1048 So these two parameters double as Boolean flags for whether the signal is
1049 required as part of the transaction, and as values for time-out of the transaction if
1050 the signal is not received.

1051 The specification of a business transaction may require each one of these
1052 signals independently of whether the other is required. If one is not required, it is
1053 actually not allowed. Therefore there is a finite set of combinations.

1054 These could then be given pattern names and business process definition tools
1055 could facilitate the application of named patterns to actual business transaction
1056 definitions.

1057 UMM provides a set of already specified transaction patterns for re-use.

1058

1059 6.5.3 Parameters required for CPP/CPA

1060

1061 The specification schema provides parameters that can be used to specify
1062 certain levels of security and reliability. The specification schema provides these
1063 parameters in general business terms.

1064 These parameters are generic requirements for the business process, but for
 1065 ebXML implementations, these parameters are specifically used to instruct the
 1066 CPP and CPA to require BSI and/or Delivery Channel capabilities to achieve the
 1067 specified service levels.

1068 The CPP and CPA translate these into parameters of two kinds.

1069 One kind of parameter determines Delivery Channel selection, that is the
 1070 selection of the transport facility and transport characteristics, depending on
 1071 security and reliability parameters.

1072 The other kind of parameter determines the service level or capability of the BSI
 1073 itself.

1074

1075 6.5.4 Delivery Channel selection parameters:

1076

1077 6.5.4.1 Document security:

1078 Each business document being transported, even if many are
 1079 collected in the same message, can be specified as:

Parameter	Delivery Channel requirement
<i>isConfidential.</i>	The information entity is encrypted so that unauthorized parties cannot view the information
<i>isTamperProof.</i>	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.
<i>isAuthenticated.</i>	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.

1080

1081 6.5.4.2 Document Flow security:

1082

1083 Each message can be specified to require secure transport.

1084

Parameter	Delivery Channel requirement
IsSecureTransportRequired	This means that Business Documents transferred must be treated as per parameters <i>isConfidential</i> ,

	<i>isTamperProof</i> , <i>isAuthenticated</i> , above
--	--

1085

1086

1087

IsSecureTransportRequired can have three values:

1088

“Transient” means *isConfidential*, *isTamperProof*, *isAuthenticated*, on the wire.

1089

1090

“Persistent” means *isConfidential*, *isTamperProof*, *isAuthenticated*, until delivered into the application space.

1091

1092

“No” means no security required

1093

1094

1095

1096

1097

1098

1099

1100

The value of *isSecureTransportRequired* and the document level values of *isConfidential*, *isTamperProof*, *isAuthenticated* combine to produce the value that calls for the strongest security. E.g. if *isSecureTransportRequired* = ‘Transient’, then all documents will be treated as *isConfidential*, *isTamperProof*, *isAuthenticated* on the wire even if their individual settings are ‘no’. Likewise a document that requires security will be treated with that security, even if *isSecureTransportRequired* is ‘no’.

1101 6.5.5 Reliability

1102

1103

A parameter at the transaction level states whether guaranteed delivery required.

1104

Parameter	Delivery Channel requirement
IsGuaranteedDeliveryRequired	This means that Business Documents transferred guaranteed to be delivered

1105

1106

This is a parameter in the business transaction

1107

1108 6.5.6 Synchronous or Asynchronous

1109

Need proper definition possibly in terms of blocking

Parameter	Delivery Channel requirement
isSynchronous	Synchronous response

1110

1111

1112

1113

A business transaction may be implemented as either a synchronous or an asynchronous flow of control between the two activities. The specification of synchronous vs. asynchronous is

1114 part of the interaction pattern specification for a business
1115 transaction.

1116 A partner role that initiates an asynchronous business transaction
1117 does not need to receive any business signals. A partner role that
1118 initiates a synchronous business transaction must be able to
1119 receive business signals and must block until the flow of control is
1120 returned. This should not preclude the initiation and execution of
1121 multiple concurrent business transactions, however.

1122

1123 6.5.7 BSI service level parameters:

1124

1125 6.5.7.1 Action security

1126

1127 Each request or response may be sent by a variety of individuals,
1128 representatives or automated systems associated with a business
1129 partner. There may be cases where trading partners have more
1130 than one ebXML-capable business service interface, representing
1131 different levels of authority. In such a case, the parties may
1132 establish rules regarding which interfaces or authors may be
1133 confidently relied upon as speaking for the enterprise.

1134 In order to invoke those rules, a party may specify
1135 `IsAuthorizationRequired` on a requesting or and responding
1136 activity accordingly, with the result that [the activity] will only be
1137 processed as valid if the party interpreting it successfully matches
1138 the stated identity of the activity's [Authorized Role] to a list of
1139 allowed values previously supplied by that party.

Parameter	BSI requirement
<code>IsAuthorizationRequired</code>	Must validate identity of originator against a list of authorized originators

1140

1141 `IsAuthorizationRequired` is specified on the requesting and
1142 responding activity accordingly.

1143

1144 6.5.7.2 Non-Repudiation

1145

1146 Trading partners may wish to conduct legally enforceable
1147 business transactions over ebXML. Parties may elect to use non-
1148 repudiation protocols in order to generate documentation that
1149 would assist in the enforcement of the contractual obligation in
1150 court, in the case that the counterparty later attempts to repudiate
1151 its ebXML messages.

1152 There are two kinds of non-repudiation protocol available under
1153 this document.

1154 One imposes a duty on each party to save copies of all [Business
1155 Documents], each on their own side, i.e., requestor saves his
1156 request, responder saves his response. This is the
1157 isNonRepudiationRequired in the requesting or responding
1158 activity. It is logically equivalent to a request that the other trading
1159 partner maintain an audit trail.

1160 The other requires the responder to send a signed copy of the
1161 receipt, which the requestor then saves. This is the
1162 isNonRepudiationOfReceiptRequired in the requesting business
1163 activity.

1164 NonRepudiationOfReceipt is tied to the ReceiptAcknowledgement,
1165 in that it requires it to be signed. So one cannot have
1166 NonRepudiationOfReceipt without ReceiptAcknowledgement
1167 required, and the timeToAcknowledgeReceipt applies to both, i.e.
1168 if NonRepudiationOfReceipt is true and a signed receipt is not
1169 returned within timeToAcknowledgeReceipt, the transaction is null
1170 and void.

Parameter	BSI requirement
isNonRepudiationRequired	Must save audit trail of messages it sends
isNonRepudiationOfReceiptRequired	Must save audit trail of receiptAcknowledgements

1171

1172

1173 **6.6 Run time transaction semantics**

1174 The ebXML concept of a business transaction and the semantics behind it are
1175 central to predictable, enforceable commerce. It is expected that any Business
1176 Service Interface (BSI) will be capable of managing a transaction according to
1177 these semantics.

1178 Therefore, the Business Service Interface (BSI), or any software that implements
1179 one role in an ebXML collaboration needs at minimum to be able to support the
1180 following transaction semantics:

- 1181 1. Detection of the opening of a transaction
- 1182 2. Detection of transfer of control
- 1183 3. Detection of successful completion of a transaction
- 1184 4. Detection of failed completion of a transaction
 - 1185 a. Detection of time-outs
 - 1186 b. Detection of exceptions

- 1187 5. Notification of failure
 1188 6. Receipt of notification of failure
 1189 7. Rollback upon failure (note this is the independent responsibility of each
 1190 role, it is not a co-coordinated roll-back, there are no 2-phase commits in
 1191 ebXML)

1192 ebXML does not specify how these transaction semantics is implemented but it is
 1193 assumed that any Business Service Interface (BSI) will be able to support these
 1194 basic transaction semantics at runtime. If either party cannot provide full support,
 1195 then the requirements may be relaxed as overrides in the CPP/CPA.

1196 The following sections discuss the two causes of failure: Time-outs and
 1197 Exceptions. When either one happens, it is the responsibility of the two roles to
 1198 do the necessary roll-back, and to exit the transaction. The responsibilities of the
 1199 two roles differ slightly and are described in each of the sections below.
 1200 Generally, if a failure happens at the responding role, the responding role will
 1201 send an exception signal to the requesting role, and both parties will exit the
 1202 current transaction. If a failure happens at the requesting role, the requesting role
 1203 will exit the current transaction and in a separate transaction notify the
 1204 responding role about the failure. This way the flow of control within a transaction
 1205 is always unambiguous and finite.

1206

1207 6.6.1 Timeouts

1208

1209 Since all business transactions must have a distinct time boundary, there
 1210 are time-out parameters associated with each of the response types. If
 1211 the time-out occurs before the required response arrives, the transaction
 1212 is null and void.

1213

1214 Here are the time-out parameters relative to the three response types:

1215

Response required	Parameter Name	Meaning of timeout
Receipt acknowledgement	timeToAcknowledgeReceipt	The time a responding role has to acknowledge receipt of a business document.
Acceptance Acknowledgement (Non-substantive)	timeToAcknowledgeAcceptance	The time a responding role has to non-

		substantively acknowledge business acceptance of a business document.
Substantive Response	TimeToPerform	The time a responding role has to substantively acknowledge business acceptance of a business document.

1216

1217

1218

1219

A time-out parameter must be specified whenever a requesting partner expects one or more responses to a business document request. A requesting partner must not remain in an infinite wait state.

1220

1221

1222

1223

The time-out value for each of the time-out parameters is absolute i.e. not relative to each other. All timers start when the requesting business document is sent. The timer values must comply with the well-formedness rules for timer values.

1224

1225

A responding partner simply terminates if a timeout is thrown. This prevents responding business transactions from hanging indefinitely.

1226

1227

A requesting partner terminates if a timeout is thrown and then sends a notification of failure to the responder as part of a separate transaction.

1228

1229

1230

1231

1232

1233

When the time to perform an activity equals the time to acknowledge receipt or the time to acknowledge business acceptance then the highest priority time out exception must be used when the originator provides a reason for revoking their original business document offer. The time to perform exception is lower priority than both the time to acknowledge receipt and the time to acknowledge business acceptance.

1234 6.6.2 Exceptions

1235

1236

1237

1238

1239

Under all normal circumstances the response message and/or the time-outs determine the success or failure of a business transaction. However the business processing of the transaction can go wrong at either the responding or the requesting role.

1240 6.6.3 ControlException

1241

1242 A *ControlException* signals an error condition in the management of a
1243 business transaction. This business signal is asynchronously returned to
1244 the initiating activity that originated the request. This exception must
1245 terminate the business transaction. These errors deal with the
1246 mechanisms of message exchange such as verification, validation,
1247 authentication and authorization and will occur up to message
1248 acceptance. Typically the rules and constraints applied to the message
1249 will have only dealt with structure, syntax and message element values.

1250 6.6.4 Business Protocol Exceptions (a.k.a. ProcessException)

1251 A *ProcessException* signals an error condition in a business activity. This
1252 business signal is asynchronously returned to the initiating role that
1253 originated the request. This exception must terminate the *business*
1254 *transaction*. These errors deal with the mechanisms that process the
1255 *business transaction* and will occur after message verification and
1256 validation. Typically the rules and constraints applied to the message will
1257 deal the semantics of message elements and the validity of the request
1258 itself and the content is not valid with respect to a responding role's
1259 business rules. This type of exception is usually generated after an
1260 *AcceptanceAcknowledgement* has been returned.

1261 A business protocol exception terminates the business transaction. The
1262 following are business protocol exceptions.

- 1263 • Negative acknowledgement of receipt. The structure/schema of a
1264 message is invalid.
- 1265 • Negative acknowledgement of acceptance. The business rules
1266 are violated.
- 1267 • Performance exceptions. The requested business action cannot
1268 be performed.
- 1269 • Sequence exceptions. The order or type of a business document
1270 or business signal is incorrect.
- 1271 • Syntax exceptions. There is invalid punctuation, vocabulary or
1272 grammar in the business document or business signal.
- 1273 • Authorization exceptions. Roles are not authorized to participate in
1274 the business transaction.
- 1275 • Business process control exceptions. Business documents are not
1276 signed for non-repudiation when required.

1277 A responding role that throws a business protocol exception signals the
1278 exception back to the requesting role and then terminates the business
1279 transaction. A requesting role that throws a business protocol exception
1280 terminates the transaction and then sends as a separate transaction a
1281 notification revoking the offending business document request. The
1282 requesting role does not send a business exception signal to the
1283 responding role.

1284 If any business exceptions (includes negative receipt and acceptance
1285 acknowledgements) are signaled then the business transaction must
1286 terminate.

1287 **6.7 Runtime Collaboration Semantics**

1288 The ebXML collaboration semantics contain a number of relationships between
1289 multiparty collaborations and binary collaborations, between recursive layers of
1290 binary collaborations, and choreographies among transactions in binary
1291 collaborations. It is anticipated that over time BSI software will evolve to the point
1292 of monitoring and managing the state of a collaboration, similar to the way a BSI
1293 today is expected to manage the state of a transaction. For the immediate future,
1294 such capabilities are not expected and not required.

1295 **6.8 Where the ebXML Specification Schema May Be 1296 Implemented**

1297 The ebXML Specification Schema should be used wherever software is being
1298 specified to perform a role in an ebXML business collaboration. Specifically The
1299 ebXML Specification Schema is intended to provide the business process and
1300 document specification for the formation of ebXML trading partner Collaboration
1301 Protocol Profiles and Agreements.

1302 However, the Specification Schema may be used to specify any electronic
1303 commerce collaboration. It may also be used for non-commerce collaborations,
1304 for instance in defining transactional collaborations among non-profit
1305 organizations or internally in enterprises.

1306 **7 UML Element Specification**

1307

1308 In the following we will review all the specification elements in the UML
1309 Specification Schema, grouped as follows:

- 1310 • Business Collaborations
 - 1311 ○ Multiparty
 - 1312 ○ Binary
- 1313 • Business Transactions
- 1314 • Document Flow
- 1315 • Choreography

1316

1317 **7.1 Business Collaborations**

1318

1319 7.1.1 MultiPartyCollaboration

1320 A Multiparty Collaboration is a synthesis of Binary Collaborations.
 1321 A Multiparty Collaboration consists of a number of Business
 1322 Partner Roles each playing roles in binary collaborations with
 1323 each other.

1324 **Tagged Values:**

1325 *NONE*

1326 **Associations:**

1327 *partners* A multiparty collaboration has two or more
 1328 BusinessPartnerRoles

1329 **Wellformedness Rules:**

1330 All multiparty collaborations must be synthesized from binary
 1331 collaborations

1332

1333 7.1.2 BusinessPartnerRole

1334 A BusinessPartnerRole is the role played by a business partner in
 1335 a MultiPartyCollaboration. A BusinessPartnerRole performs at
 1336 most one Authorized Roles in each of the Binary Collaborations
 1337 that make up the Multiparty Collaboration.

1338 **Tagged Values:**

1339 *name.* The name of the roles played by partner in the
 1340 overall multiparty business collaboration, e.g.
 1341 customer or supplier

1342 **Associations:**

1343 *performers.* The Authorized Roles performed by a partner in
 1344 the binary business collaboration.

1345 *transitions* The transitions (managed by this
 1346 BusinessPartnerRole) between activities across
 1347 binary collaborations

1348 **Wellformedness Rules:**

1349 A partner must not perform both roles in a given business
 1350 activity.

1351

1352 7.1.3 Performs

1353 Performs is an explicit modeling of the relationship between a
 1354 BusinessPartnerRole and the Roles it plays. This specifies the use
 1355 of an Authorized Role within a multiparty collaboration.

1356 **Tagged Values:**

1357

1358 *NONE*1359 **Associations:**1360 *performedBy* An instance of Performs is performed by only
1361 one BusinessPartnerRole1362 *role* Performs is the use of an AuthorizedRole within
1363 a multiparty collaboration1364 **Wellformedness Rules:**1365 For every Performs performing an AuthorizedRole there must be
1366 a Performs that performs the opposing
1367 AuthorizedRole, otherwise the MultiParty
1368 Collaboration is not complete.

1369

1370 **7.1.4 AuthorizedRole**

1371

1372 An Authorized Role is a role that is authorized to send the request
1373 or response, e.g. the buyer is authorized to send the request for
1374 purchase order, the seller is authorized to send the acceptance of
1375 purchase order.1376 **Tagged Values:**1377 *name.* The name of the role within the business
1378 transaction1379 **Associations:**1380 *performers* An AuthorizedRole may be used by one or more
1381 performers, i.e. Business Partner Roles in a
1382 multiparty collaboration1383 *from* An AuthorizedRole may be the initiator in a
1384 business activity1385 *to* An AuthorizedRole may be the responder in a
1386 business activity1387 *collaboration* An AuthorizedRole may be in only one
1388 BinaryCollaboration1389 **Wellformedness Rules:**1390 An AuthorizedRole may not be both the requestor and the
1391 responder in a business transaction1392 An AuthorizedRole may not be both the initiator and the
1393 responder in a binary business collaboration

1394

1395 7.1.5 BinaryCollaboration

1396 A Binary Collaboration defines a protocol of interaction between
1397 two authorized roles.

1398 A Binary Collaboration is a choreographed set of states among
1399 collaboration roles. The activities of performing business
1400 transactions or other collaborations are a kind of state.

1401 A Binary Collaboration choreographs one or more business
1402 transaction activities between two roles.

1403 A Binary Collaboration is not an atomic transaction and should not
1404 be used in cases where transaction rollback is required.

1405 **Tagged Values:**

1406 *timeToPerform.* The time allowed to complete the binary
1407 collaboration

1408 *requires* A description of a state external to this
1409 collaboration/transaction that is required before
1410 this collaboration/transaction to conclude.

1411 *resultsIn* A description of a state that does not exist
1412 before the execution of this
1413 collaboration/transaction but will exist as a
1414 result of the execution of this
1415 collaboration/transaction.

1416 *beginsWhen* A description of an event external to the
1417 collaboration/transaction that normally causes
1418 this collaboration/transaction to commence.

1419 *endsWhen* A description of an event external to this
1420 collaboration/transaction that normally causes
1421 this collaboration/transaction to conclude.

1422 *Pattern* The optional name of the pattern that this binary
1423 collaboration is based on

1424 **Associations:**

1425 *role* A binary collaboration consists of two authorized
1426 roles

1427 *states* A binary collaboration consists of one or more
1428 states, some of which are 'static', and some of
1429 which are action states

1430 *usedBy* A binary collaboration may be used within
1431 another binary collaboration via a collaboration
1432 activity

1433 *transitions* The transitions between activities in this binary
1434 collaboration

1435 **Wellformedness Rules:**

1436 NONE

1437

1438 **7.1.6 BusinessActivity**

1439

1440

A business activity is an action state within a binary collaboration. It is the super type for BusinessTransactionActivity and CollaborationActivity, specifying the activity of performing a transaction or another binary collaboration respectively.

1441

1442

1443

1444

Tagged Values:

1445

name.

The name of the activity within the binary collaboration

1446

1447

Associations:

1448

from

The initiating role

1449

to

The responding role

1450

Wellformedness Rules:

1451

NONE

1452

1453 **7.1.7 BusinessTransactionActivity**

1454

A business transaction activity defines the use of a business transaction within a binary collaboration.

1455

1456

A business transaction activity is a business activity that executes a specified business transaction. The business transaction activity can be executed more than once if the *isConcurrent* property is *true*.

1457

1458

1459

1460

Tagged Values:

1461

timeToPerform. Both partners agree to perform a business transaction activity within a specific duration. The originating partner must send a failure notification to a responding partner on timeout. A responding partner simple terminates its activity. The time to perform is the duration from the time a business transaction activity initiates the first business transaction until there is a transition back to the initiating business transaction activity. Both partners agree that the business signal document or business action document specified as the document to return within the time to perform is the "Acceptance Document" in an on-line offer/acceptance contract formation process.

1462

1463

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

1475

1476

isConcurrent.

If the business transaction activity is concurrent then more than one business transaction can be open at one time. If the business transaction

1477

1478

1479 activity is not concurrent then only one business
 1480 transaction activity can be open at one time.

1481 *isLegallyBinding* The Business Transaction performed by this
 1482 activity is intended by the trading parties to be
 1483 binding. Default value is True.

1484 *isSynchronous* The Business Transaction is intended to be
 1485 performed by this activity in a synchronous
 1486 manner

1487

1488 **Associations:**

1489 *uses.* The business transaction activity executes
 1490 (uses) exactly one business transaction.

1491 **Wellformedness Rules:**

1492 NONE

1493

1494 **7.1.8 CollaborationActivity**

1495 A collaboration activity defines the use of one binary collaboration
 1496 within another.

1497 A collaboration activity is the activity of performing a binary
 1498 collaboration within another binary collaboration.

1499 **Tagged Values:**

1500 *NONE (other than inherited)*

1501 **Associations:**

1502 *uses.* A collaboration activity uses exactly one binary
 1503 collaboration

1504 **Wellformedness Rules:**

1505 A binary collaboration may not re-use itself

1506

1507

1508 **7.2 Business Transactions**

1509

1510 **7.2.1 BusinessTransaction**

1511 A business transaction is a set of business information and
 1512 business signal exchanges amongst two commercial partners that
 1513 must occur in an agreed format, sequence and time period. If any
 1514 of the agreements are violated then the transaction is terminated
 1515 and all business information and business signal exchanges must

1516		be discarded. Business Transactions can be formal as in the
1517		formation of on-line offer/acceptance commercial contracts and
1518		informal an in the distribution of product announcements.
1519		Tagged Values:
1520	<i>name</i>	The name of the Business Transaction.
1521	<i>isSecureTransportRequired.</i>	
1522		Both partners must agree to exchange business
1523		information using a secure Delivery Channel.
1524		The following security controls ensure that
1525		business document content is protected against
1526		unauthorized disclosure or modification and that
1527		business services are protected against
1528		unauthorized access. This is a point-to-point
1529		security requirement. Note that this requirement
1530		does not protect business information once it is
1531		off the network and inside an enterprise. The
1532		following are requirements for secure transport
1533		Delivery Channels.
1534		Authenticate sending role identity – Verify the
1535		identity of the sending role (employee or
1536		organization) that is initiating the role interaction
1537		(authenticate).
1538		Authenticate receiving role identity – Verify
1539		the identity of the receiving role (employee or
1540		organization) that is receiving the role
1541		interaction.
1542		Verify content integrity – Verify the integrity of
1543		the content exchanged during the role
1544		interaction i.e. check that the content has not
1545		been altered by a 3 rd party.
1546		Maintain content confidentiality –
1547		Confidentiality ensures that only the intended,
1548		receiving role can read the content of the role
1549		interaction
1550	<i>isGuaranteedDeliveryRequired.</i>	Both partners must agree to use
1551		a transport that guarantees delivery
1552	<i>requires</i>	A description of a state external to this
1553		collaboration/transaction that is required before
1554		this collaboration/transaction to conclude.
1555	<i>resultsIn</i>	A description of a state that does not exist
1556		before the execution of this
1557		collaboration/transaction but will exists as a
1558		result of the execution of this
1559		collaboration/transaction.
1560	<i>beginsWhen</i>	A description of an event external to the
1561		collaboration/transaction that normally causes
1562		this collaboration/transaction to commence.

1563 *endsWhen* A description of an event external to this
 1564 collaboration/transaction that normally causes
 1565 this collaboration/transaction to conclude.

1566 *pattern* The optional name of the pattern that this
 1567 business transaction is based on

1568 **Wellformedness Rules:**

1569 NONE

1570

1571

1572 **7.2.2 Business Action**

1573 A Business Action is an abstract super class. Business Action, is
 1574 the holder of attributes that are common to both Requesting
 1575 Business Activity and Responding Business Activity.

1576 **Tagged Values:**

1577 *IsAuthorizationRequired* If a partner role needs authorization to
 1578 request a business action or to respond to a
 1579 business action then the sending partner role
 1580 must sign the business document exchanged
 1581 and the receiving partner role must validate this
 1582 business control and approve the authorizer. A
 1583 responding partner must signal an authorization
 1584 exception if the sending partner role is not
 1585 authorized to perform the business activity. A
 1586 sending partner must send notification of failed
 1587 authorization if a responding partner is not
 1588 authorized to perform the responding business
 1589 activity.

1590 *IsNonRepudiationRequired* If non-repudiation of origin and
 1591 content is required then the business activity
 1592 must store the business document in its original
 1593 form for the duration mutually agreed to in a
 1594 trading partner agreement. A responding partner
 1595 must signal a business control exception if the
 1596 sending partner role has not properly delivered
 1597 their business document. A requesting partner
 1598 must send notification of failed business control
 1599 if a responding partner has not properly
 1600 delivered their business document.

1601 *isNonRepudiationOfReceiptRequired.*

1602 Both partners agree to mutually verify receipt of a
 1603 requesting business document and that the receipt must
 1604 be non-reputable. A receiving partner must send
 1605 notification of failed business control (possibly revoking a
 1606 contractual offer) if a responding partner has not
 1607 properly delivered their business document.

1608

1609 Non-repudiation of receipt provides the following audit
 1610 controls.
 1611 **Verify responding role identity** (authenticate) – Verify
 1612 the identity of the responding role (individual or
 1613 organization) that received the requesting business
 1614 document.
 1615 **Verify content integrity** – Verify the integrity of the
 1616 original content of the business document request.

1617 *timeToAcknowledgeAcceptance* The time a receiving role has to
 1618 non-substantively acknowledge business
 1619 acceptance of a business document.

1620 *timeToAcknowledgeReceipt* The time a receiving role has to
 1621 acknowledge receipt of a business document.

1622 **Associations:**

1623 NONE

1624 **Wellformedness Rules:**

1625 NONE

1626

1627

1628 **7.2.3 RequestingBusinessActivity**

1629 A RequestingBusinessActivity is a business activity that is
 1630 performed by a role requesting commerce from another role. It
 1631 specifies the Document Flow which will carry the request.

1632 **Tagged Values:**

1633 NONE, except as inherited from Business Action

1634 **Associations:**

1635 *transaction* A requesting activity is performed in exactly one
 1636 business transaction

1637 *documentFlow* A responding activity sends at most one
 1638 Document Flow

1639

1640 **Wellformedness Rules:**

1641 NONE

1642

1643 **7.2.4 RespondingBusinessActivity**

1644 A RespondingBusinessActivity is a business activity that is
 1645 performed by a role responding to another business role's request
 1646 for commerce.

1647 It specifies the Document Flow which will carry the request.

1648 There may be multiple possible response Document Flows
 1649 defined, but only one of them will happen during an actual
 1650 transaction instance.

1651 **Tagged Values:**

1652 NONE, except as inherited from Business Action

1653 **Associations:**

1654 *transaction* A responding activity is performed in exactly one
 1655 business transaction

1656 *documentFlow* A responding activity sends at most one
 1657 Document Flow

1658

1659 **Wellformedness Rules:**

1660 NONE

1661

1662 **7.3 Document Flow**

1663

1664 **7.3.1 Document Flow**

1665 A Document Flow is what conveys business information between
 1666 the two roles in a business transaction. One Document Flow
 1667 conveys the request from the requesting role to the responding
 1668 role, and another Document Flow conveys the response (if any)
 1669 from the responding role back to the requesting role.

1670

1671 **Tagged Values:**

1672 *isSuccess* An expression whose evaluation results in
 1673 TRUE or FALSE as the determination of
 1674 whether this DocumentFlow should be
 1675 considered the successful initiation or
 1676 conclusion of a BusinessTransaction.

1677 **Associations:**

1678 *requesting* A Document Flow is sent by at most one
 1679 requesting activity

1680 *responding* A Document Flow is sent by at most one
 1681 responding activity

1682 *documentType* A Document Flow contains only one Document

1683 *attachment* A Document Flow contains an optional set of
 1684 attachments to the document

1685

Wellformedness Rules:

1686

A Document Flow cannot be sent by both a requesting and a responding activity

1687

1688

1689

1690

1691

1692

1693 **7.3.2 DocumentType**

1694

DocumentType is a generic name of a document. The definition of the document can be found in the associated Schema. Associated with the document can optionally be a set of attachments

1695

1696

1697

Tagged Values:

1698

name The name of the document type

1699

Associations:

1700

schema A document type is in at most one schema

1701

documentFlow A document type can be in multiple document flows

1702

1703

attachment A document type can specify many attachments

1704

Wellformedness Rules:

1705

NONE

1706 **7.3.3 Schema**

1707

A Schema is a collection of Document Definitions. The Schema is usually external to the process specification, and is referenced with a URI. An additional reference is to where the logical model is for the Documents in the Schema. Typically this would be an ebXML core component context model.

1708

1709

1710

1711

1712

Tagged Values:

1713

Name The name of the schema

1714

location Reference to an external source of the schema definition

1715

1716

logicalModel Reference is to where the logical model is

1717

1718

Associations:

1719

documentType A schema defines many document types

1720 **Wellformedness Rules:**

1721 NONE

1722

1723 **7.3.4 Attachment**1724 Attachment is an optional attachment to a DocumentType in a
1725 Document Flow1726 **Tagged Values:**1727 *name* The name of the attachment1728 *mimeType* Defines the valid MIME (Multipurpose Internet
1729 Mail Extensions) type of this Attachment1730 *spec* A reference to an external source of description
1731 of this attachment.1732 **Associations:**1733 *documentFlow* An Attachment is in exactly one document flow1734 *documentType* An Attachment is of at most one document type.
1735 If it is not of a defined document type, the mime
1736 type and spec will be the only indication of its
1737 type1738 **Wellformedness Rules:**1739 An attachment is always related to the primary document an a
1740 document flow

1741

1742

1743 **7.4 Choreography within Collaborations.**

1744

1745

1746 **7.4.1 BusinessState**1747 A business state is any state that a binary collaboration can be in.
1748 Some business states are a snapshot right after or right before an
1749 activity, others are action states that denote the state of being in
1750 an activity.1751 **Tagged Values:**1752 *none*1753 **Associations:**1754 *collaboration* A business state belongs to only one binary
1755 collaboration

1756 *entering* A transition that reflects entry into this state
 1757 *exiting* A transition that reflects exiting from this state

1758 **Wellformedness Rules:**

1759 NONE

1760

1761 **7.4.2 Transition**

1762 A transition is a transition between two business states in a binary
 1763 collaboration.

1764 Choreography is expressed as transitions between business
 1765 states

1766 **Tagged Values:**

1767 *onInitiation* Is used to specify this is a nested
 1768 BusinessTransactionActivity and a second
 1769 transaction is performed before returning to the
 1770 this transaction to send the response back to the
 1771 original requestor

1772 *guardCondition* A reference to the status of the previous
 1773 transaction. A fixed value of Success,
 1774 BusinessFailure, TechnicalFailure, or AnyFailure

1775 *guardExpression* An expression whose evaluation results in
 1776 TRUE or FALSE to determine if this transition
 1777 should happen or not. The expression can refer
 1778 to the name or content of the document in the
 1779 most recent DocumentFlow.

1780 **Associations:**

1781 *in* The business state this transition is entering

1782 *out* The business state this transition is exiting

1783 **Wellformedness Rules:**

1784 A transition cannot enter and exit the same state

1785

1786 **7.4.3 Start**

1787 The starting state for an Binary Collaboration. A Binary
 1788 Collaboration should have at least one starting activity. If none
 1789 defined, then all activities are considered allowable entry points.

1790 **Tagged Values:**

1791 NONE

1792 **Associations:**

1793 NONE

1794		Wellformedness Rules:
1795		NONE
1796	7.4.4 TerminalState	
1797		The ending state of an binary collaboration, sub classed by
1798		success and failure
1799		Tagged Values:
1800		<i>NONE</i>
1801		Associations:
1802		<i>NONE</i>
1803		Wellformedness Rules:
1804		<i>NONE</i>
1805	7.4.5 Success	
1806		A subtype of TerminalState which defines the successful
1807		conclusion of a binary collaboration as a transition from an activity.
1808		Tagged Values:
1809		<i>NONE.</i>
1810		Associations:
1811		<i>NONE</i>
1812		Wellformedness Rules:
1813		Every activity Binary Collaboration should have at least one
1814		success
1815		
1816	7.4.6 Failure	
1817		A subtype of TerminalState which defines the unsuccessful
1818		conclusion of a binary collaboration as a transition from an activity.
1819		Tagged Values:
1820		<i>None.</i>
1821		Associations:
1822		<i>None</i>
1823		Wellformedness Rules:
1824		Every Binary Collaboration should have at least one failure

1825 7.4.7 Fork

1826 A Fork is a state with one inbound transition and multiple
 1827 outbound transitions. All activities pointed to by the outbound
 1828 transitions are assumed to happen in parallel.

1829 **Tagged Values:**

1830 *None*

1831 **Associations:**

1832 *None*

1833 **Wellformedness Rules:**

1834 *None*

1835

1836 7.4.8 Join

1837 A business state where an activity is waiting for the completion of
 1838 one or more other activities. Defines the point where previously
 1839 forked activities join up again.

1840 **Tagged Values:**

1841 *waitForAll* Boolean value indicating if this Join state should
 1842 wait for all incoming transitions to complete. If
 1843 TRUE, wait for all, if False proceed on first
 1844 incoming transition.

1845 **Associations:**

1846 *None*

1847 **Wellformedness Rules:**

1848 *None*

1849

1850 7.4.9 Guard

1851 The condition under which a transition may happen.

1852 **Tagged Values:**

1853 *Precondition* A condition referring to the status of the previous
 1854 document flow: Success, or the outcome of the
 1855 previous transaction activity: BusinessFailure,
 1856 TechnicalFailure

1857 **Associations:**

1858 *Transition* *The transition(s) that this guard governs*

1859 **Wellformedness Rules:**

1860 Guards must refer only to the immediately prior document flows
1861 and/or the immediately prior transaction activity
1862

1863 **7.5 Definition and Scope**

1864 The ebXML Specification Schema should be used wherever software is being
1865 specified to perform a role in an ebXML binary collaboration. Specifically The
1866 ebXML Specification Schema is intended to provide the business process and
1867 document specification for the formation of a trading partner Collaboration
1868 Protocol Profile and Agreement. A set of specification rules have been
1869 established to properly constrain the expression of a business process and
1870 information model in a way that can be directly incorporated into a trading partner
1871 Collaboration Protocol Profile and Agreement.

1872 **7.6 Collaboration Specification Rules**

1873 The following rules are used to constrain the values of the parameters of the
1874 elements in the Specification Schema.

1875

1876 **7.6.1 Well-formedness Rules**

1877 The following well-formedness rules apply :

1878 *Business Transaction*

- 1879 [0] If non-repudiation is required then the input or returned business
1880 document must be a tamper-proofed entity.
- 1881 [1] If authorization is required then the input business document and
1882 business signal must be an authenticated or a tamper proofed secure
1883 entity.
- 1884 [2] The time to acknowledge receipt must be less than the time to
1885 acknowledge acceptance if both properties have values.
1886
1887 $timeToAcknowledgeReceipt < timeToAcknowledgeAcceptance$
- 1888 [3] If the time to acknowledge acceptance is null then the time to perform
1889 an activity must either be equal to or greater than the time to
1890 acknowledge receipt.
- 1891 [4] The time to perform a transaction cannot be null if either the time to
1892 acknowledge receipt or the time to acknowledge acceptance is not
1893 null.
- 1894 [5] If non-repudiation of receipt is required then the time to acknowledge
1895 receipt cannot be null.
- 1896 [6] The time to acknowledge receipt, time to acknowledge acceptance
1897 and time to perform cannot all be zero.

- 1898 [7] If non-repudiation is required at the requesting business activity, then
1899 there must be a responding business document.
- 1900 [8] The time to acknowledge receipt, time to acknowledge acceptance
1901 and time to perform properties must be specified for both the
1902 requesting and responding business activities and they must be
1903 equal.
- 1904 *RequestingBusinessActivity*
- 1905 [9] There must be one input transition whose source state vertex is an
1906 initial pseudo state.
- 1907 [10] There must be one output transition whose target state vertex is a
1908 final state specifying the state of the machine when the activity is
1909 successfully performed.
- 1910 [11] There must be one output transition whose target state vertex is a
1911 final state specifying the state of the machine when the activity is NOT
1912 successfully performed due to a process control exception.
- 1913 [12] There must be one output transition whose target state vertex is a
1914 final state specifying the state of the machine when the activity is NOT
1915 successfully performed due to a business process exception.
- 1916 [13] There must be one output Document Flow that in turn is the input
1917 to a responding business activity.
- 1918 [14] There must be zero or one output Document Flow from a
1919 requesting that in turn is the input to the requesting business activity.
- 1920 *RespondingBusinessActivity*
- 1921 [15] There must be one input transition from a Document Flow that in
1922 turn has one input transition from a requesting business activity.
- 1923 [16] There must be zero or one output transition to an Document Flow
1924 that in turn has an output transition to a requesting business activity.
- 1925 *Business Collaboration*
- 1926 [17] A Business Partner Role cannot provide both the initiating and
1927 responding roles of the same business transaction activity.

1928 8 Specification Schema – (DTD)

1929 In this section we describe the DTD version of the Specification Schema. This
1930 discussion includes

- 1931 • A listing of the DTD itself
- 1932 • A table listing all the elements found in the DTD with definitions and
1933 parent/child relationships
- 1934 • A table listing all the attributes found in the DTD with definitions and
1935 parent element relationships
- 1936 • A table listing all the elements found in the DTD, each with a cross
1937 reference to the corresponding class in the UML version of the
1938 specification schema
- 1939 • An example using the DTD
- 1940 • Rules about namespaces

1941 Additionally, following this section, a section will describe common modeling
1942 elements, including data types, and DTDs for common business signals. And a
1943 section after that describes the production rules we intend to follow for creating
1944 XML documents from a model instance against the UML specification schema.

1945 8.1 DTD

1946 This is the Specification Schema DTD:
1947

```

1948 <? version="1.0" encoding="UTF-8"?>
1949 <!--===== -->
1950 <!-- Editor: Kurt Kanaskie (Lucent Technologies) -->
1951 <!-- Version: Version .99 -->
1952 <!-- Updated: March 20, 2001 -->
1953 <!-- -->
1954 <!-- Public Identifier: -->
1955 <!-- "-//ebXML//DTD Process Specification ver .99//EN" -->
1956 <!-- -->
1957 <!-- Purpose: The ebXML Specification DTD provides a standard -->
1958 <!-- framework by which business systems may be -->
1959 <!-- configured to support execution of business -->
1960 <!-- transactions. It is based upon prior UN/CEFACT -->
1961 <!-- work, specifically the metamodel behind the -->
1962 <!-- UN/CEFACT Unified Modeling Methodology (UMM) defined -->
1963 <!-- in the N90 specification. -->
1964 <!-- -->
1965 <!-- The Specification Schema supports the specification -->
1966 <!-- of Business Transactions and the choreography of -->
1967 <!-- Business Transactions into Business Collaborations. -->
1968 <!-- Notes: -->
1969 <!-- time periods are represented using ISO 8601 format -->

```

```

1970 <!--      (e.g. P2D for 2 Days, P2H30M for 2 Hours 30 Minute  -->
1971 <!-- ===== -->
1972
1973 <!ELEMENT EbXmlProcessSpecification (Documentation*, Include*,
1974 Package*)>
1975 <!ATTLIST EbXmlProcessSpecification
1976     name CDATA #REQUIRED
1977     version CDATA #REQUIRED
1978     uuid CDATA #REQUIRED
1979 >
1980 <!ELEMENT Documentation (#PCDATA)>
1981 <!ATTLIST Documentation
1982     uri CDATA #IMPLIED
1983 >
1984
1985 <!ELEMENT Include EMPTY>
1986 <!ATTLIST Include
1987     name CDATA #REQUIRED
1988     version CDATA #REQUIRED
1989     uuid CDATA #REQUIRED
1990     uri CDATA #REQUIRED
1991 >
1992 <!ELEMENT Package (Documentation*,
1993     (Package | BinaryCollaboration |
1994 MultiPartyCollaboration | BusinessTransaction | Schema)*)>
1995 <!ATTLIST Package
1996     name CDATA #REQUIRED
1997 >
1998 <!-- Model requires 2 Authorized roles -->
1999 <!ELEMENT BinaryCollaboration (Documentation*, AuthorizedRole,
2000 AuthorizedRole,
2001     (Documentation* | Start | Transition |
2002 Success | Failure | Fork | Join | BusinessTransactionActivity |
2003 CollaborationActivity)*)>
2004 <!ATTLIST BinaryCollaboration
2005     name CDATA #REQUIRED
2006     pattern CDATA #IMPLIED
2007     beginsWhen CDATA #IMPLIED
2008     endsWhen CDATA #IMPLIED
2009     requires CDATA #IMPLIED
2010     resultsIn CDATA #IMPLIED
2011     timeToPerform CDATA #IMPLIED
2012 >
2013 <!-- fromBusinessState and toBusinessState are the names of a
2014 BusinessTransactionActivity or CollaborationActivity
2015 both are logical targets for from/to -->
2016 <!-- "guardExpression" is an expression that results in a boolean
2017 true or false -->
2018 <!ELEMENT Transition (Documentation*)>
2019 <!ATTLIST Transition

```

```
2020         onInitiation (true | false) "false"
2021         fromBusinessState CDATA #IMPLIED
2022         toBusinessState CDATA #IMPLIED
2023         guardCondition (Success | BusinessFailure |
2024 TechnicalFailure | AnyFailure ) #IMPLIED
2025         guardExpression CDATA #IMPLIED
2026     >
2027 <!--ELEMENT Start EMPTY-->
2028 <!--ATTLIST Start
2029         toBusinessState CDATA #REQUIRED
2030     >
2031 <!--ELEMENT Success EMPTY-->
2032 <!--ATTLIST Success
2033         fromBusinessState CDATA #REQUIRED
2034         guardCondition (Success | BusinessFailure |
2035 TechnicalFailure | AnyFailure ) #IMPLIED
2036         guardExpression CDATA #IMPLIED
2037     >
2038 <!--ELEMENT Failure EMPTY-->
2039 <!--ATTLIST Failure
2040         fromBusinessState CDATA #REQUIRED
2041         guardCondition (Success | BusinessFailure |
2042 TechnicalFailure | AnyFailure ) #IMPLIED
2043         guardExpression CDATA #IMPLIED
2044     >
2045
2046 <!--ELEMENT Fork EMPTY-->
2047 <!--ATTLIST Fork
2048         name CDATA #REQUIRED
2049     >
2050 <!--ELEMENT Join EMPTY-->
2051 <!--ATTLIST Join
2052         name CDATA #REQUIRED
2053         waitForAll (true | false) "true"
2054     >
2055 <!-- fromAuthorizedRole and toAuthorizedRole are fully qualified
2056 names -->
2057 <!--ELEMENT BusinessTransactionActivity (Documentation*)-->
2058 <!--ATTLIST BusinessTransactionActivity
2059         name CDATA #REQUIRED
2060         businessTransaction CDATA #REQUIRED
2061         fromAuthorizedRole CDATA #REQUIRED
2062         toAuthorizedRole CDATA #REQUIRED
2063         isConcurrent (true | false) "false"
2064         isLegallyBinding (true | false) "true"
2065         isSynchronous (true | false) "false"
2066         timeToPerform CDATA #IMPLIED
2067     >
2068 <!--ELEMENT CollaborationActivity (Documentation*)-->
2069 <!--ATTLIST CollaborationActivity
```

```
2070     name CDATA #REQUIRED
2071     fromAuthorizedRole CDATA #REQUIRED
2072     toAuthorizedRole CDATA #REQUIRED
2073     binaryCollaboration CDATA #REQUIRED
2074 >
2075 <!ELEMENT BusinessTransaction (Documentation*,
2076 RequestingBusinessActivity, RespondingBusinessActivity)>
2077 <!ATTLIST BusinessTransaction
2078     name CDATA #REQUIRED
2079     pattern CDATA #IMPLIED
2080     beginsWhen CDATA #IMPLIED
2081     endsWhen CDATA #IMPLIED
2082     isSecureTransportRequired (no | persistent | transient)
2083     "no"
2084     isGuaranteedDeliveryRequired (true | false) "false"
2085     requires CDATA #IMPLIED
2086     resultsIn CDATA #IMPLIED
2087 >
2088 <!ELEMENT RequestingBusinessActivity (Documentation*,
2089 DocumentFlow)>
2090 <!ATTLIST RequestingBusinessActivity
2091     name CDATA #REQUIRED
2092     isAuthorizationRequired (true | false) "false"
2093     isIntelligibleCheckRequired (true | false) "false"
2094     isNonRepudiationReceiptRequired (true | false) "false"
2095     isNonRepudiationRequired (true | false) "false"
2096     timeToAcknowledgeAcceptance CDATA #IMPLIED
2097     timeToAcknowledgeReceipt CDATA #IMPLIED
2098 >
2099 <!ELEMENT RespondingBusinessActivity (Documentation*,
2100 DocumentFlow*)>
2101 <!ATTLIST RespondingBusinessActivity
2102     name CDATA #REQUIRED
2103     isAuthorizationRequired (true | false) "false"
2104     isIntelligibleCheckRequired (true | false) "false"
2105     isNonRepudiationReceiptRequired (true | false) "false"
2106     isNonRepudiationRequired (true | false) "false"
2107     timeToAcknowledgeAcceptance CDATA #IMPLIED
2108     timeToAcknowledgeReceipt CDATA #IMPLIED
2109 >
2110 <!ELEMENT Schema (Documentation*, DocumentType*)>
2111 <!ATTLIST Schema
2112     name CDATA #REQUIRED
2113     location CDATA #IMPLIED
2114     logicalModel CDATA #IMPLIED
2115 >
2116 <!ELEMENT DocumentType (Documentation*)>
2117 <!ATTLIST DocumentType
2118     name CDATA #REQUIRED
2119 >
```

```
2120 <!ELEMENT DocumentFlow (Documentation*, Attachment*)>
2121 <!-- "isSuccess" is an expression that results in a boolean true
2122 or false -->
2123 <!-- documentType is the fully qualified name -->
2124 <!ATTLIST DocumentFlow
2125     documentType CDATA #REQUIRED
2126     isSuccess CDATA #REQUIRED
2127     isAuthenticated (true | false) "false"
2128     isConfidential (true | false) "false"
2129     isTamperProof (true | false) "false"
2130 >
2131 <!ELEMENT Attachment (Documentation*)>
2132 <!ATTLIST Attachment
2133     name CDATA #REQUIRED
2134     mimeType CDATA #REQUIRED
2135     documentType CDATA #IMPLIED
2136     specification CDATA #IMPLIED
2137     isConfidential (true | false) "false"
2138     isTamperProof (true | false) "false"
2139     isAuthenticated (true | false) "false"
2140 >
2141 <!ELEMENT AuthorizedRole (Documentation*)>
2142 <!ATTLIST AuthorizedRole
2143     name CDATA #REQUIRED
2144 >
2145 <!ELEMENT MultiPartyCollaboration (Documentation*,
2146 BusinessPartnerRole*)>
2147 <!ATTLIST MultiPartyCollaboration
2148     name CDATA #REQUIRED
2149 >
2150 <!ELEMENT BusinessPartnerRole (Documentation*, Performs*,
2151 Transition*)>
2152 <!ATTLIST BusinessPartnerRole
2153     name CDATA #REQUIRED
2154 >
2155 <!-- authorizedRole is the fully qualified name
2156 BinaryCollaboration@name/AuthorizedRole@name -->
2157 <!ELEMENT Performs (Documentation*)>
2158 <!ATTLIST Performs
2159     authorizedRole CDATA #REQUIRED
2160 >
2161
```

2162 **8.2 Documentation for the DTD**

2163
2164 This section will document the DTD. The DTD has been derived from the UML
2165 model. The correlation between the UML classes and DTD elements will be
2166 shown separately later in this document.

2167

2168

a. Attachment

2169

XML Element Name: Attachment

2170

DTD Declaration:

2171

```
<!ELEMENT Attachment (Documentation*)>
```

2172

```
<!ATTLIST Attachment
```

2173

```
  name CDATA #REQUIRED
```

2174

```
  mimeType CDATA #REQUIRED
```

2175

```
  Spec CDATA #REQUIRED
```

2176

```
  isConfidential (true | false) "false"
```

2177

```
  isTamperProof (true | false) "false"
```

2178

```
  isAuthenticated (true | false) "false">
```

2179

Definition:

2180

An optional attachment to a DocumentType in a DocumentFlow.

2181

2182

Parent Elements:

2183

- DocumentFlow

2184

Attributes:

Attribute Name	Definition	Default Value
isAuthenticated	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.	false {true, false}
isConfidential	The information entity is encrypted so that unauthorized parties cannot view the information	false {true, false}
isTamperProof	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.	false Valid values {true, false}
mimeType	Defines the valid MIME (Multipurpose Internet Mail Extensions) type of this Attachment	Required input. Example: 'application/pdf'
name	Defines the name of the attachment.	Required input.
Spec	A reference to an external source of description of this attachment.	Required input.

2185

2186

b. Authorized Role

2187

XML Element Name: AuthorizedRole

2188

DTD Declaration:

2189

`<!ELEMENT AuthorizedRole (Documentation*)>`

2190

`<!ATTLIST AuthorizedRole`

2191

`name CDATA #REQUIRED>`

2192

2193

Definition:

2194

An Authorized Role is a role that is authorized to send the request or response, e.g. the buyer is authorized to send the request for purchase order, the seller is authorized to send the acceptance of purchase order.

2195

2196

2197

2198

Parents:

2199

BinaryCollaboration

2200

Attributes:

Attribute Name	Definition	Default Value
----------------	------------	---------------

Name	Defines the name of the Authorized Role	Required input.
-------------	---	------------------------

2201

2202

c. Binary Collaboration

2203

XML Element Name: BinaryCollaboration

2204

DTD Declaration:

2205

`<!ELEMENT BinaryCollaboration (Documentation*,`

2206

`AuthorizedRole, AuthorizedRole,`

2207

`(Documentation* | Start | Transition | Success |`

2208

`Failure | Fork | Join | BusinessTransactionActivity |`

2209

`CollaborationActivity)*>`

2210

`<!ATTLIST BinaryCollaboration`

2211

`name ID #REQUIRED`

2212

`beginsWhen CDATA #IMPLIED`

2213

`endsWhen CDATA #IMPLIED`

2214

`requires CDATA #IMPLIED`

2215

`resultsIn CDATA #IMPLIED`

2216

`timeToPerform CDATA #IMPLIED`

2217

`>`

2218

Definition:

2219

A Binary Collaboration defines a protocol of interaction between two authorized roles.

2220

2221

A Binary Collaboration is a choreographed set of states among collaboration roles. The activities of performing business transactions or other collaborations are a kind of state.

2222

2223

2224

A Binary Collaboration choreographs one or more business transaction activities between two roles.

2225

2226

A Binary Collaboration is not an atomic transaction and should not be used in cases where transaction rollback is required.

2227

2228

2229

Parents:

2230

- Package

2231

Attributes:

2232

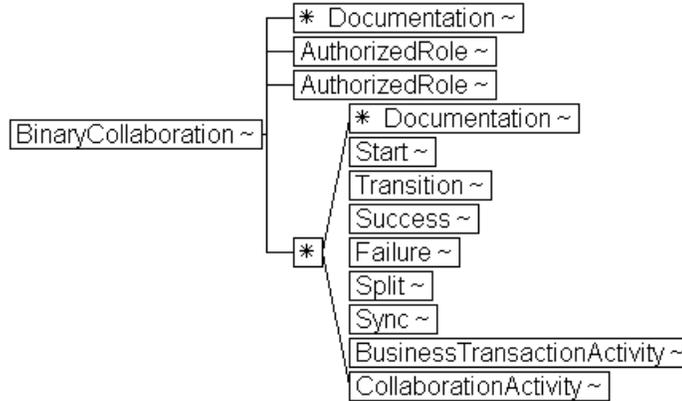
Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input.
beginsWhen	A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to commence.	Optional Input.

endsWhen	A description of an event external to this collaboration/transaction that normally causes this collaboration/transaction to conclude.	Optional Input.
requires	A description of a state external to this collaboration/transaction that is required before this collaboration/transaction to conclude.	Optional Input.
resultsIn	A description of a state that does not exist before the execution of this collaboration/transaction but will exist as a result of the execution of this collaboration/transaction.	Optional Input..
timeToPerform	The period of time, starting upon initiation of the first activity within which this entire collaboration must conclude.	Optional Input.

2233

2234
2235
2236

Hierarchical Model:



2237
2238

c: Business Partner Role

2239
2240
2241

Element Name: BusinessPartnerRole

DTD Declaration:

2242
2243
2244
2245
2246
2247

```
<!ELEMENT BusinessPartnerRole (Documentation*, Performs*,
                                Transition*)>
<!ATTLIST BusinessPartnerRole
            name          CDATA          #REQUIRED>
```

Definition:

2248
2249
2250
2251

A BusinessPartnerRole is the role played by a business partner in a MultiPartyCollaboration. A BusinessPartnerRole performs at most one Authorized Roles in each of the Binary Collaborations that make up the Multiparty Collaboration.

Parents:

2252
2253
2254

- MultiPartyCollaboration

Attributes:

2255
2256

Attribute Name	Definition	Default Value
Name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input.

2257

d. Business Transaction

2258

Element Name: BusinessTransaction

2259

Content Model:

2260

```
<!ELEMENT BusinessTransaction (Documentation*,
RequestingBusinessActivity, RespondingBusinessActivity)>
```

2261

2262

```
<!ATTLIST BusinessTransaction
```

2263

```
  name ID #REQUIRED
```

2264

```
  beginsWhen CDATA #IMPLIED
```

2265

```
  endsWhen CDATA #IMPLIED
```

2266

```
  isSecureTransportRequired (no | persistent |
                             transient) "no"
```

2267

2268

```
  isGuaranteedDeliveryRequired (true | false) false
```

2269

```
  requires CDATA #IMPLIED
```

2270

```
  resultsIn CDATA #IMPLIED>
```

2271

2272

Definition:

2273

A business transaction is a set of business information and business signal exchanges amongst two commercial partners that must occur in an agreed format, sequence and time period. If any of the agreements are violated then the transaction is terminated and all business information and business signal exchanges must be discarded. Business Transactions can be formal as in the formation of on-line offer/acceptance commercial contracts and informal as in the distribution of product announcements.

2274

2275

2276

2277

2278

2279

2280

2281

2282

2283

Parents:

2284

- Package

2285

Attributes:

2286

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input.
beginsWhen	A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to commence.	Optional Input..
endsWhen	A description of an event external to this	Optional Input.

	collaboration/transaction that normally causes this collaboration/transaction to conclude.	
isSecureTransportRequired	Any document exchanged as part of this transaction must be sent in manner that guarantees that it isConfidential, isTamperProof, and isAuthenticated	no Valid Values: {no, persistent, transient }
isGuaranteedDeliveryRequired	Delivery of every document exchanged as part of this transaction must be guaranteed	false Valid Values: {true, false}
requires	A description of a state external to this collaboration/transaction that is required before this collaboration/transaction to conclude.	Optional Input.
resultsIn	A description of a state that does not exist before the execution of this collaboration/transaction but will exists as a result of the execution of this collaboration/transaction	Optional Input.

2287

2288

e. Business Transaction Activity

2289

Element Name: BusinessTransactionActivity

2290

Content Model:

2291

`<!ELEMENT BusinessTransactionActivity (Documentation*)>`

2292

`<!ATTLIST BusinessTransactionActivity`

2293

`name CDATA #REQUIRED`

2294

`businessTransaction CDATA #REQUIRED`

2295

`fromAuthorizedRole CDATA #REQUIRED`

2296

`toAuthorizedRole CDATA #REQUIRED`

2297

`isConcurrent (true | false) "false"`

2298

`isLegallyBinding (true | false) "true"`

2299

`isSynchronous (true | false) "false"`

2300

`timeToPerform CDATA #IMPLIED>`

2301

Definition:

2302

Defines the use of a business transaction within a binary collaboration.

2303 A business transaction activity is a business activity that executes a specified
 2304 business transaction. The business transaction activity can be executed more
 2305 than once if the **isConcurrent** property is **true**.

2306 "fromAuthorizedRole" and "toAuthorizedRole" must match one of the
 2307 AuthorizedRoles in the binary collaboration. "from AuthorizedRole" will become
 2308 the requestor in the transaction, "toAuthorizedRole" will become the responder.

2309 **Parents:**

- BinaryCollaboration

2310 **Attributes:**

2311

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input.
businessTransaction	A reference, by name to the Business Transaction used by this Business Transaction Activity	Required Input.
fromAuthorizedRole	The name of the initiating role in Business Transaction Activity	Required Input.
toAuthorizedRole	Name of the responding role in Business Transaction Activity	Required Input.
timeToPerform	The time a responding role has to acknowledge business acceptance of a business document	Optional Input.
isLegallyBinding	Defines whether the business transaction as performed in this Business Transaction Activity is legally binding.	true Valid Values: {true, false}
isConcurrent	Defines whether more than one business transaction can be open during the same time period.	false Valid Values: {true, false}
isSynchronous	The Business Transaction is intended to be performed by this activity in a synchronous manner	false Valid Values: {true, false}

2312

2313 **f. Collaboration Activity**

2314 **Element Name:** CollaborationActivity

2315 **DTD Declaration:**

```

2316      <!ELEMENT CollaborationActivity (Documentation*)>
2317      <!ATTLIST CollaborationActivity
2318          name ID #REQUIRED
2319          fromAuthorizedRole CDATA #REQUIRED
2320          toAuthorizedRole CDATA #REQUIRED
2321          binaryCollaboration CDATA #REQUIRED>
2322

```

Definition:

2323 Defines the use of one binary collaboration within another.

2324 A collaboration activity is the activity of performing a binary collaboration within
2325 another binary collaboration.

2326 "fromAuthorizedRole" and "toAuthorizedRole" must match one of the
2327 AuthorizedRoles in the binary collaboration. "from AuthorizedRole" will become
2328 the initiator in the used binary collaboration, "toAuthorizedRole" will become the
2329 responder.

Parents:

- 2331 • BinaryCollaboration

2332

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input.
fromAuthorizedRole	The name of the initiating role in Collaboration Activity	Required Input
toAuthorizedRole	Name of the responding role in Collaboration Activity	Required Input.
binaryCollaboration	A reference, by name, to the Binary Collaboration used by this Collaboration Activity	Required Input.

2333

g. Documentation

2334 **Element Name:** Documentation

2335

DTD Declaration:

```

2337      <!ELEMENT Documentation (#PCDATA)>
2338      <!ATTLIST Documentation
2339          uri CDATA #IMPLIED>
2340

```

Definition:

2342 Defines user documentation for any element. Must be the first element of its
2343 container. Documentation can be either inline PCDATA and/or a URI to where
2344 more complete documentation is to be found

2345

Parents:

- 2346 • AuthorizedRole
- 2347 • BinaryCollaboration
- 2348 • BusinessPartnerRole
- 2349 • BusinessTransaction
- 2350 • BusinessTransactionActivity
- 2351 • CollaborationActivity
- 2352 • DocumentFlow
- 2353 • DocumentType
- 2354 • EbXmlProcessSpecification
- 2355 • MultiPartyCollaboration
- 2356 • Package
- 2357 • Performs
- 2358 • RequestingBusinessActivity
- 2359 • RespondingBusinessActivity
- 2360 • Schema
- 2361 • Transition

Attributes:

2363

Attribute Name	Definition	Default Value
uri	Defines the URI (Uniform Resource Identifier) where external documentation is located.	No Default Value. Valid URI is required.

2364

2365

h. Document Flow

2366

Element Name: DocumentFlow

2367

Content Model:

2368

```
<!ELEMENT DocumentFlow (Documentation*, DocumentType, Attachment*)>
```

2369

2370

```
<!ATTLIST DocumentFlow
```

2371

```
  isSuccess CDATA #REQUIRED
```

2372

```
  isAuthenticated (true | false) "false"
```

2373

```
  isConfidential (true | false) "false"
```

2374

```
  isTamperProof (true | false) "false">
```

2375

2376

Definition:

2377

A Document Flow is what conveys business information between the two roles in a business transaction. One Document Flow conveys the request from the requesting role to the responding role, and another Document Flow conveys the response (if any) from the responding role back to the requesting role.

2378

2379

2380

2381

Parents:

2382

- RequestingBusinessActivity

2383

- RespondingBusinessActivity

2384

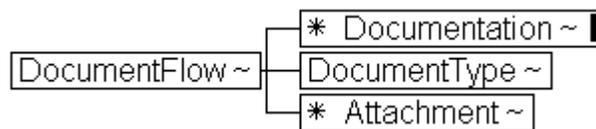
2385

Attributes:

Attribute Name	Definition	Default Value
isSuccess	An expression whose evaluation results in TRUE or FALSE as the determination of whether this DocumentFlow should be considered the successful initiation or conclusion of a BusinessTransaction.	Required Input.
isAuthenticated	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.	false Valid Values: {true, false}
isConfidential	The information entity is encrypted so that unauthorized parties cannot view the information	false Valid Values: {true, false}
isTamperProof	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.	false Valid Values: {true, false}

2386

2387

Hierarchical View:

2388

2389

i. DocumentType

2390

Element Name: DocumentType

2391

DTD Declaration:

2392

`<!ELEMENT DocumentType (Documentation*) >`

2393

`<!ATTLIST DocumentType`

2394

`name CDATA #REQUIRED >`

2395

Definition:

2396 DocumentType is a generic name of a document. The definition of the document
 2397 can be found in the associated Schema. Associated with the document can
 2398 optionally be a set of attachments

2399 Specifies the document type in the form of "SchemaName/DocumentType"

2400 **Parents:**

- 2401 • Attachment
- 2402 • Schema

2403 **Attributes:**
 2404
 2405

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input

2406

2407

j. ebXMLProcessSpecification

2408

Element Name: ebXMLProcessSpecification

2409

DTD Declaration:

2410

```
<!ELEMENT EbXMLProcessSpecification (Documentation*,
2411                                     Include*, Package*) >
```

2411

2412

```
<!ATTLIST EbXMLProcessSpecification
2413         name          CDATA      #REQUIRED
2414         version       CDATA      #REQUIRED
2415         uuid          CDATA      #REQUIRED >
```

2413

2414

2415

2416

Definition:

2417

Root element of a process specification document that has a globally unique identity.

2418

2419

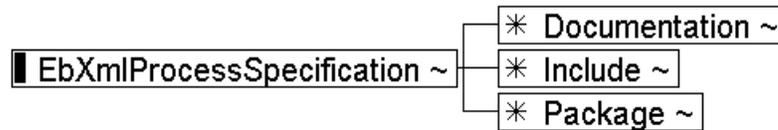
Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input
uuid	Universally unique identifier.	No Default Value
version	Version of the specification.	No Default Value

2420

2421

Hierarchical Model:



2422

2423

2424

k. Failure

2425

Element Name: Failure

2426

DTD Declaration:

2427

`<!ELEMENT Failure EMPTY>`

2428

`<!ATTLIST Failure`

2429

`fromBusinessState CDATA #REQUIRED`

2430

`guardCondition (Success | BusinessFailure |`

2431

`TechnicalFailure | AnyFailure) #IMPLIED`

2432

`guardExpression CDATA #IMPLIED`

2433

`>`

2434

Definition:

2435

Defines the unsuccessful conclusion of a binary collaboration as a transition from an activity.

2436

2437

Parents:

2438

- BinaryCollaboration

2439

Attributes:

Attribute Name	Definition	Default Value
fromBusinessState	The name of the prior activity.	Required Input.
guardCondition	A reference to the status of the previous transaction. A fixed value of success, businessFailure, technicalFailure, or anyFailure	Optional Input. Valid Values: {Success, BusinessFailure, TechnicalFailure, AnyFailure}
guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition to success should happen or not. The expression can refer to the name or content of the document in the most recent DocumentFlow	Optional Input

2440

2441

2442

l. Include

2443

Element Name: Include

2444

DTD Declaration:

```

2445      <!ELEMENT Include EMPTY >
2446      <!ATTLIST Include
2447          name CDATA #REQUIRED
2448          version CDATA #REQUIRED
2449          uuid CDATA #REQUIRED
2450          uri CDATA #REQUIRED >
2451

```

Definition:

2452 Includes another process specification document and merges that specification
 2453 with the current specification. Any elements of the same name and in the same
 2454 name scope must have exactly the same specification except that packages may
 2455 have additional content.

2456 Documents are merged based on name scope. A name in an included package
 2457 will be indistinguishable from a name in the base document.

Parents:

- 2459 • EbXmlProcessSpecification

2460

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input
uri	Uniform Resource Indicator.	No Default Value
uuid	Universally unique identifier.	No Default Value
version	Version of the included specification.	No Default Value

2461

2462

m. MultiParty Collaboration2464 **Element Name:** MultiPartyCollaboration2465 **DTD Declaration:**

```

2466      <!ELEMENT MultiPartyCollaboration (Documentation*,
2467          BusinessPartnerRole+) >
2468      <!ATTLIST MultiPartyCollaboration
2469          name CDATA #REQUIRED >
2470

```

Definition:

2471 A Multiparty Collaboration is a synthesis of Binary Collaborations. A Multiparty
 2472 Collaboration consists of a number of Business Partner Roles each playing roles
 2473 in binary collaborations with each other.

Parents:

- 2475 • Package

2476

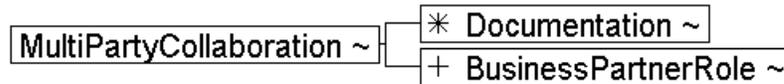
Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input

2477

2478

Hierarchical Model:



2479

2480

n. Package

2481

Element Name: Package

2482

DTD Declaration:

2483

```

    <!ELEMENT Package (Documentation*, (Package |
    2484 BinaryCollaboration |
    2485 MultiPartyCollaboration |
    2486 BusinessTransaction)*) >
  
```

2484

2485

2486

2487

```

    <!ATTLIST Package
    2488 name CDATA #REQUIRED >
  
```

2488

2489

Definition:

Defines a hierarchical name scope containing reusable elements.

2490

Parents:

2491

- ebXMLProcessSpecification
- package

2492

2493

2494

2495

2496

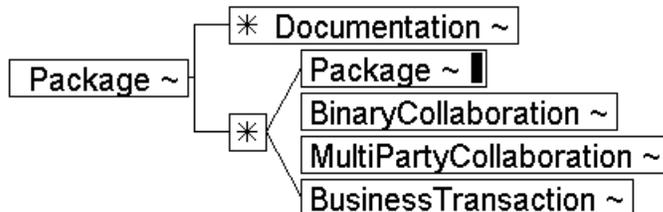
Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input

2497

2498

Hierarchical Model:



2499

2500

o. Performs

2501

Element Name: Performs

2502

DTD Declaration:

2503

`<!ELEMENT Performs (Documentation*)>`

2504

`<!ATTLIST Performs`

2505

`authorizedRole CDATA #REQUIRED`

2506

`>`

2507

2508

Definition:

2509

Performs is an explicit modeling of the relationship between a BusinessPartnerRole and the Roles it plays. This specifies the use of an Authorized Role within a multiparty collaboration.

2510

2511

2512

Parents:

2513

- BusinessPartnerRole

2514

Attributes:

Attribute Name	Definition	Default Value
authorizedRole	Name of the Authorized that will be performed by the business partner role, qualified with the name of the binary collaboration	Required Input

2515

2516

2517

2518

p. Requesting Business Activity

2519

Element Name: RequestingBusinessActivity

2520

DTD Declaration:

2521

```
<!ELEMENT RequestingBusinessActivity (Documentation*,
DocumentFlow) >
```

2522

2523

```
<!ATTLIST RequestingBusinessActivity
```

2524

```
name CDATA #REQUIRED
```

2525

```
isAuthorizationRequired (true | false) "false"
```

2526

```
isIntelligibleCheckRequired (true | false) "false"
```

2527

```
isNonRepudiationReceiptRequired (true | false) "false"
```

2528

```
isNonRepudiationRequired (true | false) "false"
```

2529

```
timeToAcknowledgeAcceptance CDATA #IMPLIED
```

2530

```
timeToAcknowledgeReceipt CDATA #IMPLIED>
```

2531

Definition:

2532

A RequestingBusinessActivity is a business activity that is performed by a role requesting commerce from another role. It specifies the Document Flow which will carry the request.

2533

2534

2535

Parents:

2536

- BusinessTransaction

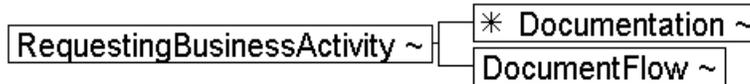
2537

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input
isAuthorizationRequired	Must validate identity of originator against a list of authorized originators	false Valid Values: {true, false}
isIntelligibleCheckRequired	Receiving partner role must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt	false Valid Values: {true, false}
isNonRepudiationReceiptRequired	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt.	false Valid Values: {true, false}
isNonRepudiationRequired	Requires the sending parties to save copies of the transacted documents before sending	false Valid Values: {true, false}
timeToAcknowledgeAcceptance	The time a responding role has to non-substantively acknowledge business acceptance of a business document.	No default value.
timeToAcknowledgeReceipt	The time a responding role has to acknowledge receipt of a business document	No default value.

2538

2539

Hierarchical Model:

2540

2541

2542

q. Responding Business Activity

2543

Element Name: RespondingBusinessActivity

2544

DTD Declaration:

2545

```
<!ELEMENT RespondingBusinessActivity (Documentation*,
DocumentFlow) >
```

2546

2547

```
<!ATTLIST RespondingBusinessActivity
```

2548

```
  name CDATA #REQUIRED
```

2549

```
  isAuthorizationRequired (true | false) "false"
```

2550

```
  isIntelligibleCheckRequired (true | false) "false"
```

2551

```
  isNonRepudiationReceiptRequired (true | false) "false"
```

2552

```
  isNonRepudiationRequired (true | false) "false"
```

2553

```
  timeToAcknowledgeAcceptance CDATA #IMPLIED
```

2554

```
  timeToAcknowledgeReceipt CDATA #IMPLIED>
```

2555

Definition:

2556

A RespondingBusinessActivity is a business activity that is performed by a role responding to another business role's request for commerce.

2557

2558

It references the Document Flow which will carry the request.

2559

There may be multiple possible response Document Flows defined, but only one of them will happen during an actual transaction instance.

2560

2561

Parents:

2562

- BusinessTransaction

2563

Attributes:

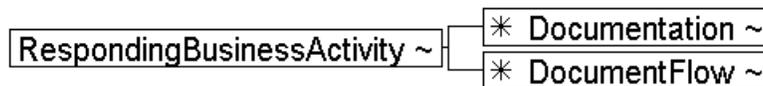
Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input
isAuthorizationRequired	Must validate identity of originator against a list of authorized originators	false Valid Values: {true, false}
isIntelligibleCheckRequired	Receiving partner role must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of	false Valid Values: {true, false}

	receipt	
isNonRepudiationReceiptRequired	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt.	false Valid Values: {true, false}
isNonRepudiationRequired	Requires the sending parties to save copies of the transacted documents before sending	false Valid Values: {true, false}
timeToAcknowledgeAcceptance	The time a responding role has to non-substantively acknowledge business acceptance of a business document.	No default value.
timeToAcknowledgeReceipt	The time a responding role has to acknowledge receipt of a business document	No default value.

2564

2565

2566

Hierarchical Model:

2567

2568

2569

r. Schema

2570

Element Name:

2571

DTD Declaration:

2572

```
<!ELEMENT Schema (Documentation*, DocumentType*)>
```

2573

```
<!ATTLIST Schema
```

2574

```
  name CDATA #REQUIRED
```

2575

```
  location CDATA #IMPLIED
```

2576

```
  logicalModel CDATA #IMPLIED>
```

2577

Definition:

2578

A Schema is a collection of Document Definitions. The Schema is usually external to the process specification, and is referenced with a URI. An additional reference is to where the logical model is for the Documents in the Schema. Typically this would be an ebXML core component context model.

2579

2580

2581

2582

Parents:

2583

- Package

2584

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input
location	Reference to an external source of the schema definition	No default value.
logicalModel	Reference is to where the logical model is	No default value.

2585

2586

Attributes:

2587

2588

s. Fork

2589

Element Name: Fork

2590

DTD Declaration:

2591

```
<!ELEMENT Fork EMPTY >
```

2592

```
<!ATTLIST Fork
```

2593

```
name CDATA #REQUIRED >
```

2594

Definition:

2595

A Fork is a state with one inbound transition and multiple outbound transitions.

2596

All activities pointed to by the outbound transitions are assumed to happen in parallel.

2597

2598

Parents:

2599

- BinaryCollaboration

2600

Attributes:

2601

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input

2602

2603

t. Start

2604

Element Name: Start

2605

DTD Declaration:

2606

<!ELEMENT Start EMPTY >

2607

<!ATTLIST Start

2608

toBusinessState CDATA #REQUIRED >

2609

Definition:

2610

The starting state for an Binary Collaboration. A Binary Collaboration should have at least one starting activity. If none defined, then all activities are considered allowable entry points.

2611

2612

2613

Parents:

2614

- BinaryCollaboration

2615

Attributes:

Attribute Name	Definition	Default Value
toBusinessState	Name of an activity that is an allowed entry point for collaboration.	Required Input

2616

2617

2618

u. Success

2619

Element Name: Success

2620

DTD Declaration:

2621

<!ELEMENT Success EMPTY>

2622

<!ATTLIST Success

2623

fromBusinessState CDATA #REQUIRED

2624

guardCondition (Success | BusinessFailure |

2625

TechnicalFailure | AnyFailure) #IMPLIED

2626

guardExpression CDATA #IMPLIED

2627

>

2628

2629

Definition:

2630

Defines the successful conclusion of a binary collaboration as a transition from an activity.

2631

2632

Parents:

2633

- BinaryCollaboration

2634

Attributes:

Attribute Name	Definition	Default Value
fromBusinessState	The name of the prior activity.	Required Input.
guardCondition	A reference to the status of the previous transaction. A fixed value of success, businessFailure, technicalFailure, or anyFailure	Optional Input. Valid Values: {Success, BusinessFailure, TechnicalFailure, AnyFailure}

guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition to success should happen or not. The expression can refer to the name or content of the document in the most recent DocumentFlow	Optional Input
------------------------	--	----------------

2635

2636

v. Join State

2637

Element Name: Join

2638

DTD Declaration:

2639

```
<!ELEMENT Join EMPTY >
```

2640

```
<!ATTLIST Join
```

2641

```
name CDATA #REQUIRED >
```

2642

Definition:

2643

A business state where an activity is waiting for the completion of one or more other activities. Defines the point where previously forked activities join up again.

2644

2645

Parents:

2646

- BinaryCollaboration

2647

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input
waitForAll	Boolean value indicating if this Join state should wait for all incoming transitions to complete. If TRUE, wait for all, if False proceed on first incoming transition.	false Valid Values: {true, false}

2648

2649

w. Transition

2650

ELEMENT Name: Transition

2651

DTD Declaration:

2652

```
<!ELEMENT Transition (Documentation*)>
```

2653

```
<!ATTLIST Transition
```

2654

```
onInitiation (true | false) "false"
```

2655

```
fromBusinessState CDATA #IMPLIED
```

2656

```
toBusinessState CDATA #IMPLIED
```

2657

```
guardCondition (Success | BusinessFailure |
```

2658

```
TechnicalFailure | AnyFailure ) #IMPLIED
```

2659

```
guardExpression CDATA #IMPLIED
```

2660

```
>
```

2661
2662
2663
2664
2665
2666
2667
2668

Definition:

A transition is a transition between two business states in a binary collaboration.

Choreography is expressed as transitions between business states

Parents:

- BinaryCollaboration
- BusinessPartnerRole

Attributes:

Attribute Name	Definition	Default Value
onInitiation	Is used to specify this is a nested BusinessTransactionActivity and a second transaction is performed before returning to the this transaction to send the response back to the original requestor	false Valid Values: {true, false}
fromBusinessState	The name of the state transitioned from	No default value.
toBusinessState	The name of the state transitioned to	No default value.
guardCondition	A reference to the status of the previous transaction. A fixed value of success, businessFailure, technicalFailure, or anyFailure	Optional Input. Valid Values: {Success, BusinessFailure, TechnicalFailure, AnyFailure}
guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition should happen or not. The expression can refer to the name or content of the document in the most recent DocumentFlow	Optional Input

2669
2670
2671
2672
2673
2674

2675 **8.3 XML to UML cross-reference**

2676
2677
2678

The following is a table that references the XML element names in the DTD to their counterpart classes in the UML specification schema.

2679

XML Element	UML Class
Attachment	Attachment
AuthorizedRole	AuthorizedRole
Binary Collaboration	Binary Collaboration
BusinessPartner Role	BusinessPartner Role
Business Transaction Activity	Business Transaction Activity
Business Transaction	Business Transaction
Responding BusinessActivity	Responding BusinessActivity
Requesting BusinessActivity	Requesting BusinessActivity
Collaboration Activity	Collaboration Activity
DocumentFlow	DocumentFlow
Documentation	None (Should be added)
ebXml Process Specification	(From Package model: ebXml Process Specification)
Failure	Failure
Include	(From Package model: Include)
MultiParty Collaboration	MultiParty Collaboration
Package	(From Package model: Package)
Performs	Performs
Schema	Schema
Fork	Fork

Start	Start
Success	Success
Join	Join
Transition	Transition

2680

2681

2682

2683

The following classes in the UML specification schema are abstract, and do not have an element equivalent in the DTD. Only their concrete subtypes are in the DTD:

2684

- BusinessState

2685

- TerminalState

2686

- BusinessActivity

2687

- BusinessAction

2688

- DocumentSecurity

2689

2690

The following classes in the UML specification schema are in the DTD represented not by elements but by attributes in other elements:

2691

- DocumentType (attribute of DocumentFlow and of Schema)

2692

2693

2694

2695 **8.4 Scoped Name Reference**

2696

2697

2698

The structure of ebXML process specifications encourages re-use. An ebXMLProcessSpecification can include another ebXMLProcessSpecification by reference.

2699

2700

2701

2702

In addition the contents of a ebXMLprocessSpecification can be arranged in a recursive package structure. The ebXMLprocessSpecification is a package container, so it can contain packages within it. Package in itself is also a package container, so it can contain further packages within it.

2703

Packages function as namespaces as per below.

2704

2705

2706

Finally a Package, at any level can have PackageContent. Types of PackageContent are BusinessTransaction, BinaryCollaboration, MultiPartyCollaboration.

2707

2708

PackageContent are always uniquely named within a package. Lower level elements a uniquely named within their parent PackageContent.

2709

2710

2711

Each PackageContent type is a built-in context provider for the core components Logical Model for the Business Document definitions referenced by this ebXMLprocessSpecification.

2712

Within a ebXMLprocessSpecification the following applies to naming:

2713 Specification elements reference other specification elements by name. EbXML
 2714 specification element names are all contained within a name scope, usually a
 2715 package. The set of packages describes a hierarchical name space, much like a
 2716 directory structure.

2717 The name of the element defined by its "name" attribute is the "simple name" of
 2718 the element. The simple name is sufficient to reference that element within the
 2719 same name scope or any parent name scope. Simple names may only contain
 2720 the characters: a—z, A..Z, 0..9, "_".

2721 A name scope that contains another named element is referred to as the "parent"
 2722 of that named element and the contained element is the "child" of the parent
 2723 scope.

2724 The "current" name scope is the one from which the scoped name reference is
 2725 made.

2726 A simple name is "in scope", that is can be resolved, if it is in the current name
 2727 scope or a parent name scope.

2728 To access elements in other name scopes the name reference must be qualified.

2729 A name qualifier shall precede the simple name with a slash ("/") character
 2730 separating the two names. The qualifying name shall specify the simple name of
 2731 the scope in which the simple name may be found. Since a package also has a
 2732 name within a name scope, it can also be qualified. Thus qualified names may
 2733 reference any depth in the namespace hierarchy.

2734 The first simple name in a qualified name shall be the root. The current name
 2735 space shall be searched for the root and, if found, shall resolve that part of the
 2736 name. If the root is not found in the current name scope the parent scope shall
 2737 be searched, recursively, until the root is found. If the scoped name starts with
 2738 "/" the root namespace shall be the one defined by EbXMLProcessSpecification.

2739 Examples of scoped names:

2740 "Foo" (Simple name)

2741 "Billing/invoice" (Name "invoice" found in "Billing" package)

2742 "/accounting/billing/foo" (name "foo" found in "billing" package found in
 2743 "accounting" package which is in the ebXMLProcessSpecification)

2744

2745 **8.5 Sample XML document against above DTD**

2746 <?xml version="1.0"?>

2747

2748 <!DOCTYPE EbXMLProcessSpecification SYSTEM

2749 "ebXMLSpecificationDTD099_1.dtd">

2750

2751 <EbXMLProcessSpecification name="GenericQuoteOrder" version="1.1" uuid="[1234-
 2752 5678-901234]">

2753 <Package name="Ordering">

```
2754 <Documentation>
2755     The OrderBT business transaction specifies that an "Order"
2756     document will initiate the transaction and that an
2757     OrderConfirmation will be a successful reply and that an
2758     "OrderDenied" document will be a failure reply (failure indicating
2759     that a business commitment was not made)
2760 </Documentation>
2761 <Schema name="ebXML1.0" location="ebXML.org"
2762 logicalModel="ebXML.org/coreComponents">
2763     <DocumentType name="OrderDT"/>
2764     <DocumentType name="OrderConfirmationDT"/>
2765     <DocumentType name="OrderDeniedDT"/>
2766     <DocumentType name="QuoteRequestDT"/>
2767     <DocumentType name="QuoteDT"/>
2768     <DocumentType name="ShippingNoticeDT"/>
2769     <DocumentType name="PaymentNoticeDT"/>
2770     <DocumentType name="WaybillDT"/>
2771     <DocumentType name="PickupReceiptDT"/>
2772     <DocumentType name="WaybillIncompleteDT"/>
2773     <DocumentType name="DeliveryReceiptDT"/>
2774 </Schema>
2775
2776 <BusinessTransaction name="OrderBT">
2777     <RequestingBusinessActivity name="OrderBA"
2778     isNonRepudiationRequired="true">
2779         <DocumentFlow documentType="OrderDT"
2780         isSuccess="true"/>
2781     </RequestingBusinessActivity>
2782     <RespondingBusinessActivity name="OrderConfirmationBA"
2783     isNonRepudiationRequired="true">
2784         <DocumentFlow documentType="OrderConfirmationDT"
2785         isSuccess="true"/>
2786         <DocumentFlow documentType="OrderDeniedDT"
2787         isSuccess="false"/>
2788     </RespondingBusinessActivity>
2789 </BusinessTransaction>
2790 <BusinessTransaction name="QuoteBT">
2791     <RequestingBusinessActivity name="QuoteRequestBA">
2792         <DocumentFlow documentType="QuoteRequestDT"
2793         isSuccess="true"/>
2794     </RequestingBusinessActivity>
2795     <RespondingBusinessActivity name="QuoteBA">
2796         <DocumentFlow documentType="QuoteDT"
2797         isSuccess="true"/>
2798     </RespondingBusinessActivity>
```

```

2799     </BusinessTransaction>
2800     <BusinessTransaction name="ShippingNoticeBT"
2801     isSecureTransportRequired="no">
2802         <RequestingBusinessActivity name="ShippingNoticeBA">
2803             <DocumentFlow documentType="ShippingNoticeDT"
2804             isSuccess="true"/>
2805         </RequestingBusinessActivity>
2806         <RespondingBusinessActivity name="ShippingNoticeBA">
2807         </RespondingBusinessActivity>
2808     </BusinessTransaction>
2809     <BusinessTransaction name="PaymentNoticeBT">
2810         <RequestingBusinessActivity name="PaymentNoticeBA">
2811             <DocumentFlow documentType="PaymentNoticeDT"
2812             isSuccess="true"/>
2813         </RequestingBusinessActivity>
2814         <RespondingBusinessActivity name="PaymentNoticeBA">
2815         </RespondingBusinessActivity>
2816     </BusinessTransaction>
2817     <BinaryCollaboration name="OrderCollaborationCO"
2818     timeToPerform="P2D">
2819         <Documentation>
2820             The "OrderCollaboration" specifies that it will start with an
2821             "OrderBT" business-transaction-activity which indicates
2822             the buy will initiate it. If the order is confirmed it will. If
2823             the order is confirmed it will proceed to a "Shipping
2824             Notice" from the sell role and then to a "PaymentNotice"
2825             from the buy role. OrderDenied from the OrderBT activity
2826             will conclude the collaboration with a failure.
2827         </Documentation>
2828         <Documentation>
2829             timeToPerform = 2 days
2830         </Documentation>
2831         <AuthorizedRole name="buyer"/>
2832         <AuthorizedRole name="seller"/>
2833         <BusinessTransactionActivity name="OrderBTA"
2834         businessTransaction="OrderBT" fromAuthorizedRole="buyer"
2835         toAuthorizedRole="seller"/>
2836         <BusinessTransactionActivity name="ShippingNoticeBTA"
2837         businessTransaction="ShippingNoticeBT"
2838         fromAuthorizedRole="seller" toAuthorizedRole="buyer"/>
2839         <BusinessTransactionActivity name="PaymentNoticeBTA"
2840         businessTransaction="PaymentNoticeBT"
2841         fromAuthorizedRole="buyer" toAuthorizedRole="seller"/>
2842     </Documentation>

```

```

2843         fromBusinessState and toBusinessState hold the name of
2844         the BusinessTransactionActivity
2845     </Documentation>
2846     <Start toBusinessState="OrderBTA"/>
2847     <Transition fromBusinessState="OrderBTA"
2848     toBusinessState="ShippingNoticeBTA"/>
2849     <Transition fromBusinessState="ShippingNoticeBTA"
2850     toBusinessState="PaymentNoticeBTA"/>
2851     <Success fromBusinessState="PaymentNoticeBTA"
2852     guardCondition="Success"/>
2853     <Failure fromBusinessState="OrderBTA"
2854     guardCondition="BusinessFailure"/>
2855 </BinaryCollaboration>
2856 <BinaryCollaboration name="QuoteOrderCollaborationCO">
2857     <Documentation>
2858         The "QuoteOrderCollaboration" starts with a QuoteBT
2859         business transaction and then proceeds to reuse the
2860         "OrderCollaboration".
2861     </Documentation>
2862     <AuthorizedRole name="buyer"/>
2863     <AuthorizedRole name="seller"/>
2864     <BusinessTransactionActivity name="QuoteBTA"
2865     businessTransaction="QuoteBT" fromAuthorizedRole="buyer"
2866     toAuthorizedRole="seller"/>
2867     <Documentation>
2868         Here we see that the name and type of an activity does not
2869         have to be the same as we use the OrderCollaboration as an
2870         activity
2871     </Documentation>
2872     <CollaborationActivity name="OrderItCA"
2873     binaryCollaboration="OrderCollaborationCO"
2874     fromAuthorizedRole="buyer" toAuthorizedRole="seller"/>
2875     <Start toBusinessState="QuoteBTA"/>
2876     <Transition fromBusinessState="QuoteBTA"
2877     toBusinessState="OrderItCA"/>
2878     <Success fromBusinessState="OrderItCA"
2879     guardCondition="Success"/>
2880     <Failure fromBusinessState="OrderItCA"
2881     guardCondition="BusinessFailure"/>
2882 </BinaryCollaboration>
2883 <MultiPartyCollaboration name="BuySellMC">
2884     <Documentation>
2885         In the BuySell multi-party collaboration we define the
2886         business partners "buyer" and "seller" which perform the
2887         "QuoteOrderCollaboration"

```

```
2888         </Documentation>
2889         <BusinessPartnerRole name="buyer">
2890             <Performs
2891                 authorizedRole="QuoteOrderCollaborationCO/buyer"/>
2892         </BusinessPartnerRole>
2893         <BusinessPartnerRole name="seller">
2894             <Performs
2895                 authorizedRole="QuoteOrderCollaborationCO/seller"/>
2896         </BusinessPartnerRole>
2897     </MultiPartyCollaboration>
2898 </Package>
2899 <!-- End of Ordering-->
2900 <Package name="Shipping">
2901     <Documentation>
2902         A sender requests shipping from a carrier
2903     </Documentation>
2904     <BusinessTransaction name="ShippingBT">
2905         <RequestingBusinessActivity name="WaybillBA "
2906             isIntelligibleCheckRequired="true">
2907             <DocumentFlow documentType="WaybillDT"
2908                 isSuccess="true"/>
2909         </RequestingBusinessActivity>
2910         <RespondingBusinessActivity name="PickupReceiptBA">
2911             <DocumentFlow documentType="PickupReceiptDT"
2912                 isSuccess="true"/>
2913             <DocumentFlow documentType="WaybillIncompleteDT"
2914                 isSuccess="false"/>
2915         </RespondingBusinessActivity>
2916     </BusinessTransaction>
2917     <BusinessTransaction name="DeliveryAcknowledgementBT">
2918         <RequestingBusinessActivity name="DeliveryReceiptBA">
2919             <DocumentFlow documentType="DeliveryReceiptDT"
2920                 isSuccess="true"/>
2921         </RequestingBusinessActivity>
2922         <RespondingBusinessActivity name="DeliveryReceiptBA">
2923         </RespondingBusinessActivity>
2924     </BusinessTransaction>
2925     <BinaryCollaboration name="ShipCollaborationCO">
2926         <AuthorizedRole name="sender"/>
2927         <AuthorizedRole name="shipper"/>
2928         <BusinessTransactionActivity name="ShippingBTA "
2929             businessTransaction="ShippingBT" fromAuthorizedRole="sender"
2930             toAuthorizedRole="shipper"/>
2931         <BusinessTransactionActivity
2932             name="DeliveryAcknowledgementBTA "
```

```

2933     businessTransaction="DeliveryAcknowledgementBT"
2934     fromAuthorizedRole="shipper" toAuthorizedRole="sender"/>
2935     <Start toBusinessState="ShippingBTA"/>
2936     <Transition fromBusinessState="ShippingBTA"
2937     toBusinessState="DeliveryAcknowledgementBTA"/>
2938     <Success fromBusinessState="DeliveryAcknowledgementBTA"
2939     guardCondition="Success"/>
2940     <Failure fromBusinessState="ShippingBTA"
2941     guardCondition="BusinessFailure"/>
2942 </BinaryCollaboration>
2943 <MultiPartyCollaboration name="BuySellShip">
2944     <Documentation>
2945         <!-- CDATA is used as example because of the & character
2946         in the text -->
2947         <![CDATA[
2948             Multipart business collaboration : Synthesizing
2949             two service interactions BuySell & Ship across
2950             three kinds of business partners "buyer" "seller" and
2951             "carrier". Note that the "seller" performs two roles
2952             within this collaboration.
2953         ]]>
2954     </Documentation>
2955     <BusinessPartnerRole name="buyer">
2956         <Performs
2957             authorizedRole="OrderCollaborationCO/buyer"/>
2958     </BusinessPartnerRole>
2959     <BusinessPartnerRole name="seller">
2960         <Performs
2961             authorizedRole="OrderCollaborationCO/seller"/>
2962         <Performs
2963             authorizedRole="ShipCollaborationCO/sender"/>
2964         <Transition
2965             fromBusinessState="OrderCollaborationCO/ShippingNotic
2966             eBTA"
2967             toBusinessState="ShipCollaborationCO/ShippingBTA"/>
2968     </BusinessPartnerRole>
2969     <BusinessPartnerRole name="carrier">
2970         <Performs authorizedRole="shipper"/>
2971     </BusinessPartnerRole>
2972 </MultiPartyCollaboration>
2973 </Package>
2974 </EbXmlProcessSpecification>
2975
2976
2977

```

2978 9 Common Modeling Elements

2979 9.1 Data typing

2980

2981

2982

2983

2984

2985

2986

XML currently has limited data typing capability for attributes and no data typing capability for elements. EbXML recognizes that data typing is a requirement for business transactions using XML. The World Wide Web Consortium (W3C) has defined a group of core data types as part of the XML Schema Specification (XML Schema Part 2: Data types, W3C Candidate Recommendation, 24 October 2000). The data types defined by the W3C will be used for global data types.

2987

2988 9.1.1 Global Data types

2989

2990

2991

2992

2993

2994

The following two charts defines the global data types that will be used for the *ebXML Business Process Specification Schema*. The first chart defines the data types that are currently available for use natively within XML DTDs. The second chart defines the proposed data types available for use with W3C XML Schema Specification.

2995

Datatypes Natively Available in DTDs	
Datatype	Definition
ID	ID represents the ID attribute type from [XML 1.0 Recommendation (Second Edition)].
IDREF	IDREF represents the IDREF attribute type from [XML 1.0 Recommendation (Second Edition)].
IDREFS	IDREFS represents the IDREFS attribute type from [XML 1.0 Recommendation (Second Edition)].
CDATA	CDATA (Character Data) represents white space normalized strings.
ENTITY	ENTITY represents the ENTITY attribute type from [XML 1.0 Recommendation (Second Edition)].
ENTITIES	ENTITIES represents the ENTITIES attribute type from [XML 1.0 Recommendation (Second Edition)].
NMTOKEN	NMTOKEN represents the NMTOKEN attribute type from [XML 1.0 Recommendation (Second Edition)].
NMTOKENS	NMTOKENS represents the NMTOKENS attribute type from [XML 1.0 Recommendation (Second Edition)].
NOTATION	NOTATION represents the NOTATION attribute type from [XML 1.0 Recommendation (Second Edition)].

2996

2997

2998

2999

3000

The table below shows the datatypes that are not natively provided in DTDs. These datatypes will be available in W3C Schema Specification, as well as the Regular Language description for XML (RELAX) schema language that has recently been submitted to ISO.

3001

3002

3003

3004

3005

Although the datatypes are not currently natively available in DTDs (native XML parsers cannot validate) processes can be used to validate the datatypes external from the XML parser.

Datatypes Not Available in DTDs	
Datatype	Definition
string	The string datatype represents character strings in XML.
boolean	The boolean datatype has the value space required to support the mathematical concept of binary-valued logic: {true, false}.
float	The float datatype corresponds to the IEEE single-precision 32-bit floating point type [IEEE 754-1985].
double	The double datatype corresponds to IEEE double-precision 64-bit floating point type [IEEE 754-1985].
decimal	The decimal datatype represents arbitrary precision decimal numbers.
timeDuration	The timeDuration datatype represents a duration of time.
recurringDuration	The recurringDuration datatype represents a specific period of time that recurs with a specific frequency, starting from a specific point in time.
binary	The binary datatype represents arbitrary binary data.
uriReference	The uriReference datatype represents a Uniform Resource Identifier (URI) Reference.
Qname	QName represents XML qualified names.
token	The token datatype represents tokenized strings.
language	The language datatype represents natural language identifiers as defined by [RFC 1766].
Name	Name represents XML Names.
NCName	NCName represents XML "non-colonized" Names.
integer	The integer datatype is derived from decimal by fixing the value of scale to be 0.
nonPositiveInteger	nonPositiveInteger is derived from integer by setting the value of maxInclusive to be 0.
negativeInteger	The negativeInteger datatype is derived from nonPositiveInteger by setting the value of maxInclusive to be -1.
long	the long datatype is derived from integer by setting the value of maxInclusive to be 9223372036854775807 and minInclusive to be -9223372036854775808. The base type of long is integer.
int	The int datatype is derived from long by setting the value of maxInclusive to be 2147483647 and minInclusive to be -2147483648. The base type of int is long.

short	short is derived from int by setting the value of maxInclusive to be 32767 and minInclusive to be -32768. The base type of short is int.
byte	The byte datatype is derived from short by setting the value of maxInclusive to be 127 and minInclusive to be -128. The base type of byte is short.
nonNegativeInteger	The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to be 0.
unsignedLong	The unsignedLong datatype is derived from nonNegativeInteger by setting the value of maxInclusive to be 18446744073709551615. The base type of unsignedLong is nonNegativeInteger.
unsignedInt	The unsignedInt datatype is derived from unsignedLong by setting the value of maxInclusive to be 4294967295. The base type of unsignedInt is unsignedLong.
unsignedShort	The unsignedShort datatype is derived from unsignedInt by setting the value of maxInclusive to be 65535. The base type of unsignedShort is unsignedInt.
unsignedByte	The unsignedByte datatype is derived from unsignedShort by setting the value of maxInclusive to be 255. The base type of unsignedByte is unsignedShort.
positiveInteger	The positiveInteger datatype is derived from nonNegativeInteger by setting the value of minInclusive to be 1.
timeInstant	The timeInstant datatype represents a specific instant of time.
time	The time datatype represents an instant of time that recurs every day.
timePeriod	The timePeriod datatype represents a specific period of time with a give start and end.
date	The date datatype represents a timePeriod that starts at midnight of a specified day and lasts until midnight the following day.
month	The month datatype represents a timePeriod that starts at midnight on the first day of the month and lasts until the midnight that ends the last day of the month.
year	The year datatype represents a timePeriod that starts at the midnight that starts the first day of the year and ends at the midnight that ends the last day of the year.
century	The century datatype represents a timePeriod that starts at the midnight that starts the first day of the century and ends at the midnight that ends that last day of the century.
recurringDate	The recurringDate datatype is a date that recurs, specifically a day of the year such as the third of May.
recurringDay	The recurringDay datatype is a day that recurs, specifically a day of the month such as the 5th of the month.

3006

3007 9.1.2 Local Datatypes

3008
3009
3010

Local datatypes used within ebXML, i.e., currency, will be developed by the ebXML Core Components Working Group.

3011 **9.2 Business signal structures**3012
3013
3014
3015
3016
3017

Since signals do not differ in structure from business transaction to business transaction, they are defined once and for all, and their definition is implied. Here are the DTD's for receiptAcknowledgment (from the RosettaNet website, courtesy of RosettaNet, and Edifecs).⁴ and for acceptanceAcknowledgement and exception.

3018 9.2.1 ReceiptAcknowledgment DTD

3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046

```
<!--
    RosettaNet XML Message Schema
    AcknowledgmentOfReceipt_MS_D02_00.dtd (31-Oct-2000
    08:26)
    This document has been prepared by Edifecs
    (http://www.edifecs.com/)
    based On the Business Collaboration Framework from
    requirements
    in conformance with the RosettaNet methodology.
-->

<!ENTITY % common-attributes "id CDATA #IMPLIED" >

<!ELEMENT ReceiptAcknowledgment (
    NonRepudiationInformation? ) >

<!ELEMENT NonRepudiationInformation (
    OriginalMessageDigest ) >

<!ELEMENT OriginalMessageDigest
    ( #PCDATA ) >
```

3047 9.2.2 AcceptanceAcknowledgement DTD

3048
3049

⁴ From RosettaNet Implementation Framework: Core Specification, Version: Release 2.00.00, 3 January 2001

```

3050      <!--
3051          ebXML XML Message Schema
3052          AcceptanceAcknowledgment_v099.dtd (19-Mar-2001 20:26)
3053          This document is a derivation of the RosettaNet
3054          ReceiptAcknowledgement signal, used by permission.
3055      -->
3056
3057      <!ENTITY % common-attributes "id CDATA #IMPLIED" >
3058
3059      <!ELEMENT AcceptanceAcknowledgment (
3060          NonRepudiationInformation? ) >
3061
3062      <!ELEMENT NonRepudiationInformation (
3063          OriginalMessageDigest ) >
3064
3065      <!ELEMENT OriginalMessageDigest
3066          ( #PCDATA ) >
3067
3068

```

9.2.3 Exception Signal DTD

```

3069
3070      <!--
3071          ebXML XML Message Schema
3072          ebxmlException_v099.dtd (19-Mar-2001 18:25)
3073          This document is a derivation of the RosettaNet
3074          Exception signal, used by permission
3075      -->
3076
3077      <!ENTITY % common-attributes "id CDATA #IMPLIED" >
3078
3079      <!ELEMENT Exception (
3080          ExceptionDescription ,
3081          GlobalExceptionTypeCode ) >
3082
3083      <!ELEMENT ExceptionDescription (
3084          errorClassification ,
3085          errorDescription ,
3086          offendingMessageComponent? ,
3087          OffendingMessageContent? ) >
3088
3089      <!ELEMENT errorClassification
3090          ( GlobalMessageExceptionCode ) >
3091
3092      <!ELEMENT GlobalMessageExceptionCode
3093          ( #PCDATA ) >
3094
3095      <!ELEMENT errorDescription
3096          ( FreeFormText ) >
3097
3098      <!ELEMENT FreeFormText
3099          ( #PCDATA ) >
3100
3101      <!ATTLIST FreeFormText
3102

```

```

3103         xml:lang CDATA #IMPLIED >
3104
3105     <!ELEMENT offendingMessageComponent
3106         ( GlobalMessageComponentCode ) >
3107
3108     <!ELEMENT GlobalMessageComponentCode
3109         ( #PCDATA ) >
3110
3111     <!ELEMENT OffendingMessageContent (
3112         XmlContentAddressingStandard ,
3113         OffendingMessageElement* ) >
3114
3115     <!ELEMENT XmlContentAddressingStandard (
3116         standardName ,
3117         standardVersion? ) >
3118
3119     <!ELEMENT standardName
3120         ( GlobalAdministeringAuthorityCode ) >
3121
3122     <!ELEMENT GlobalAdministeringAuthorityCode
3123         ( #PCDATA ) >
3124
3125     <!ELEMENT standardVersion
3126         ( VersionIdentifier ) >
3127
3128     <!ELEMENT VersionIdentifier
3129         ( #PCDATA ) >
3130
3131     <!ELEMENT OffendingMessageElement (
3132         xmlContentAddress ,
3133         errorDescription? ,
3134         ValidationConstraint? ) >
3135
3136     <!ELEMENT xmlContentAddress
3137         ( FreeFormText ) >
3138
3139     <!ELEMENT ValidationConstraint
3140         ( #PCDATA ) >
3141
3142     <!ELEMENT GlobalExceptionTypeCode
3143         ( #PCDATA ) >
3144
3145

```

3146 10 Production Rules

3147 The Specification Production rules provide the prescriptive definition necessary
3148 to translate a UML Specification Model into an XML Specification Document and
3149 the well-formed rules necessary to populate an XML Specification Document.

3150 The Specification Production rules provide the prescriptive definition necessary
3151 to translate a UML Specification Model into a XML Specification Document. The
3152 production rules are defined for concrete classes, abstract classes, aggregate
3153 associations, specialization associations and unidirectional associations.

- 3154 1. Classes are rendered as XML elements.
- 3155 2. Class attributes are rendered as XML attributes. NOTE: occurrence
3156 requirements (required vs optional) and default values for attributes are not
3157 modeled.
- 3158 3. Specialization classes (classes that inherit from another class) are rendered
3159 as XML elements including all attributes and aggregate associations from the
3160 base class. Repeated attributes are normalized to a single occurrence.
- 3161 4. Abstract classes are not rendered in the XML DTD. Abstract classes are
3162 inherited from and represent a form of collection. A class that aggregates an
3163 abstract class, essentially aggregates “any of each” of the specialization
3164 classes.
- 3165 5. An aggregate association renders the aggregated class as an XML child
3166 element with appropriate cardinality.
- 3167 6. A unidirectional association defines an attribute in the originating class of the
3168 same name as the class the association points to. This type of attribute is
3169 called a “reference attribute” and contains the name of the class it points to.
3170 The referenced class must have a “name” attribute.
- 3171 7. A class attribute data type, that has a class of the same name with stereotype
3172 <<Enumeration>> is rendered as an XML attribute enumeration. The
3173 Enumeration class does not have an explicit association.
- 3174 8. A class attribute data type (e.g. Time) that has no corresponding definition is
3175 rendered as a string. Time is formatted following ISO 8601
3176 CCYYMMDDThhmmss.sssZ.
- 3177 9. Each class is given an optional “Documentation*” element which is intended
3178 for annotation of the specification instances. This is not modeled.
- 3179
- 3180
- 3181

3182 **11 References**

3183

3184

1. UN/CEFACT Modelling Methodology (CEFACT/TMWG/N090R8E)

3185

2. RosettaNet Implementation Framework: Core Specification, Version:

3186

Release 2.00.00, 3 January 2001

3187

3188 **12 Disclaimer**

3189

The views and specification expressed in this document are those of the authors

3190

and are not necessarily those of their employers. The authors and their

3191

employers specifically disclaim responsibility for any problems arising from

3192

correct or incorrect implementation or use of this design.

3193 13 Contact Information

3194

3195 Team Leader (Of the BP team):

3196 Paul Levine

3197 Telcordia Technologies, Inc.

3198 45 Knightsbridge Road

3199 Piscataway, N.J. 08854

3200 US

3201

3202 Phone: 732-699-3042

3203 EMail: plevine@telcordia.com

3204

3205 Sub Team Lead (Of the context/MetamodelGroup) :

3206

3207 Karsten Riemer

3208 Sun Microsystems

3209 1 Network Drive

3210 Burlington, MA 01803

3211 USA

3212

3213 Phone: 781-442-2679

3214 EMail: karsten.riemer@sun.com

3215

3216 Editor (of this document):

3217

3218 Karsten Riemer

3219 Sun Microsystems

3220 1 Network Drive

3221 Burlington, MA 01803

3222 USA

3223

3224 Phone: 781-442-2679

3225 EMail: karsten.riemer@sun.com

3226

3227 **Copyright Statement**

3228 Copyright © ebXML 2000 & 2001. All Rights Reserved.

3229

3230 This document and translations of it may be copied and furnished to others, and
3231 derivative works that comment on or otherwise explain it or assist in its implementation
3232 may be prepared, copied, published and distributed, in whole or in part, without
3233 restriction of any kind, provided that the above copyright notice and this paragraph are
3234 included on all such copies and derivative works. However, this document itself may not
3235 be modified in any way, such as by removing the copyright notice or references to the
3236 Internet Society or other Internet organizations, except as needed for the purpose of
3237 developing Internet standards in which case the procedures for copyrights defined in the
3238 Internet Standards process must be followed, or as required to translate it into languages
3239 other than English.

3240

3241 The limited permissions granted above are perpetual and will not be revoked by ebXML
3242 or its successors or assigns.

3243

3244 This document and the information contained herein is provided on an
3245 "AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR
3246 IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE
3247 USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3248 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
3249 PARTICULAR PURPOSE.

3250