



Creating A Single Global Electronic Market

1

2 **ebXML Technical Architecture Specification v1.0.4**

3 **ebXML Technical Architecture Project Team**

4

5 16 February 2001

6 ***1 Status of this Document***

7

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

12

13 ***This version:***

14 ebXML_TA_v1.0.4.doc

15

16 ***Latest version:***

17 ebXML_TA_v1.0.4.doc

18

19 ***Previous version:***

20 ebXML_TA_v1.0.3.doc

21 ***2 ebXML Technical Architecture Participants***

22

23 We would like to recognize the following for their significant participation in the
24 development of this document.

25

26 Team Lead: Anders Grangard, EDI France

27

28 Editors: Brian Eisenberg, DataChannel
29 Duane Nickull, XML Global Technologies

30

31

32 Participants: Colin Barham, TIE
33 Al Boseman, ATPCO
34 Christian Barret, GIP-MDS
35 Dick Brooks, Group 8760
36 Cory Casanave, DataAccess Technologies
37 Robert Cunningham, Military Traffic Management Command, US Army
38 Christopher Ferris, Sun Microsystems
39 Peter Kacandes, Sun Microsystems
40 Kris Ketels, SWIFT

41	
42	Piming Kuo, Worldspan
43	Kyu-Chul Lee, Chungnam National University
44	Henry Lowe, OMG
45	Matt MacKenzie, XML Global Technologies
46	Melanie McCarthy, General Motors
47	Stefano Pagliani, Sun Microsystems
48	Bruce Peat, eProcessSolutions
49	John Petit, KPMG Consulting
50	Mark Heller, MITRE
51	Scott Hinkelman, IBM
52	Karsten Riemer, Sun Microsystems
53	Lynne Rosenthal, NIST
54	Nikola Stojanovic, Encoda Systems, Inc.
55	Jeff Sutor, Sun Microsystems
56	David RR Webber, XML Global Technologies
57	

57 **3 Introduction**

58

59 **3.1 Summary of Contents of Document**

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

63

64 The following conventions are used throughout this document:

- 65 • *Capitalized Italics* words are defined in the ebXML Glossary.
- 66 • [NOTES: are used to further clarify the discussion or to offer additional
- 67 suggestions and/or resources]

68

69 **EBXML TECHNICAL ARCHITECTURE SPECIFICATION V1.0.3 1**

70 1 STATUS OF THIS DOCUMENT 1

71 2 EBXML TECHNICAL ARCHITECTURE PARTICIPANTS 1

72 3 INTRODUCTION 3

73 3.1 *Summary of Contents of Document*..... 3

74 3.2 *Audience and Scope* 4

75 3.3 *Related Documents* 5

76 3.4 *Normative References* 5

77 4 DESIGN OBJECTIVES 5

78 4.1 *Problem Description & Goals for ebXML*..... 5

79 4.2 *Caveats and Assumptions* 6

80 4.3 *Design Conventions for ebXML Specifications* 6

81 5 EBXML SYSTEM OVERVIEW 7

82 6 EBXML RECOMMENDED MODELING METHODOLOGY 9

83 6.1 *Overview* 9

84 6.2 *ebXML Business Operational View* 11

85 6.3 *ebXML Functional Service View*..... 13

86 7 EBXML FUNCTIONAL PHASES 14

87 7.1 *Implementation Phase*..... 14

88 7.2 *Discovery and Retrieval Phase* 14

89 7.3 *Run Time Phase* 15

90 8 EBXML INFRASTRUCTURE 16

91 8.1 *Trading Partner Information [CPP and CPA's]* 16

92 8.1.1 *Introduction*..... 16

93 8.1.2 *CPP Formal Functionality*..... 16

94 8.1.3 *CPA Formal Functionality*..... 16

95 8.1.4 *CPP Interfaces*..... 17

96 8.1.5 *CPA Interfaces* 18

97 8.1.6 *Non-Normative Implementation Details [CPP and CPA's]* 18

98 8.2 *Business Process and Information Modeling* 19

99 8.2.1 *Introduction*..... 19

100 8.2.2 Formal Functionality..... 21

101 8.2.3 Interfaces 21

102 8.2.4 Non-Normative Implementation Details..... 22

103 8.3 Core Components and Core Library Functionality..... 23

104 8.3.1 Introduction..... 23

105 8.3.2 Formal Functionality..... 23

106 8.3.3 Interfaces 23

107 8.3.4 Non-Normative Implementation Details..... 24

108 8.4 Registry Functionality..... 25

109 8.4.1 Introduction..... 25

110 8.4.2 Formal Functionality..... 25

111 8.4.3 Interfaces 27

112 8.4.4 Non-Normative Implementation Details..... 27

113 8.5 Messaging Service Functionality..... 28

114 8.5.1 Introduction..... 28

115 8.5.2 Formal Functionality..... 30

116 8.5.3 Interfaces 30

117 8.5.4 Non-Normative Implementation Details..... 31

118 9 CONFORMANCE..... 32

119 9.1 Introduction..... 32

120 9.2 Conformance to ebXML..... 32

121 9.3 Conformance to the Technical Architecture Specification 33

122 9.4 General Framework of Conformance Testing 33

123 10.0 SECURITY CONSIDERATIONS..... 33

124 10.1 Introduction..... 34

125 DISCLAIMER..... 34

126 COPYRIGHT STATEMENT..... 34

127 APPENDIX A: EXAMPLE EBXML BUSINESS SCENARIOS 35

128 SCENARIO 1 : TWO TRADING PARTNERS SET-UP AN AGREEMENT AND RUN THE

129 ASSOCIATED EXCHANGE..... 35

130 SCENARIO 2: THREE OR MORE PARTIES SET-UP A BUSINESS PROCESS IMPLEMENTING A

131 SUPPLY-CHAIN AND RUN THE ASSOCIATED EXCHANGES 36

132 SCENARIO 3 : A COMPANY SETS UP A PORTAL WHICH DEFINES A BUSINESS PROCESS

133 INVOLVING THE USE OF EXTERNAL BUSINESS SERVICES 37

134 SCENARIO 4: THREE OR MORE TRADING PARTNERS CONDUCT BUSINESS USING SHARED

135 BUSINESS PROCESSES AND RUN THE ASSOCIATED EXCHANGES 38

137 **3.2 Audience and Scope**

138

139 This document is intended primarily for the ebXML project teams to help guide their

140 work. Secondary audiences include, but are not limited to: software implementers,

141 international standards bodies, and other industry organizations.

142

143 This document describes the underlying architecture for ebXML. It provides a high level

144 overview of ebXML and describes the relationships, interactions, and basic functionality

145 of ebXML. It SHOULD be used as a roadmap to learn: (1) what ebXML is, (2) what
146 problems ebXML solves, and (3) core ebXML functionality and architecture.

147

148 **3.3 Related Documents**

149

150 As mentioned above, other documents provide detailed definitions of the components of
151 ebXML and of their inter-relationship. They include ebXML specifications on the
152 following topics:

153

- 154 1. Requirements
- 155 2. Business Process and Information Meta Model
- 156 3. Core Components
- 157 4. Registry and Repository
- 158 5. Trading Partner Information
- 159 6. Messaging Services

160

161 These specifications are available for download at <http://www.ebxml.org>.

162

163 **3.4 Normative References**

164

165 The following standards contain provisions that, through reference in this text, constitute
166 provisions of this specification. At the time of publication, the editions indicated below
167 were valid. All standards are subject to revision, and parties to agreements based on this
168 specification are encouraged to investigate the possibility of applying the most recent
169 editions of the standards indicated below.

170

- 171 ISO/IEC 14662: Open-edi Reference Model
- 172 ISO 11179/3 Metadata Repository
- 173 ISO 10646: Character Encoding
- 174 ISO 8601:2000 Date/Time/Number Data typing
- 175 OASIS Registry/Repository Technical Specification
- 176 RFC 2119: Keywords for use in RFC's to Indicate Requirement Levels
- 177 UN/CEFACT Modeling Methodology (UMM)
- 178 W3C XML v1.0 Second Edition Specification

179 **4 Design Objectives**

180

181 **4.1 Problem Description & Goals for ebXML**

182

183 For over 25 years *Electronic Data Interchange (EDI)* has given companies the prospect
184 of eliminating paper documents, reducing costs, and improving efficiency by exchanging
185 business information in electronic form. Ideally, companies of all sizes could conduct
186 *eBusiness* in a completely ad hoc fashion, without prior agreement of any kind. But this
187 vision has not been realized with *EDI*; only large companies are able to afford to

188 implement it, and much *EDI*-enabled *eBusiness* is centered around a dominant enterprise
189 that imposes proprietary integration approaches on its *Trading Partners*.

190

191 In the last few years, the *Extensible Markup Language (XML)* has rapidly become the
192 first choice for defining data interchange formats in new *eBusiness* applications on the
193 Internet. Many people have interpreted the *XML* groundswell as evidence that "*EDI* is
194 dead" – made completely obsolete by the *XML* upstart -- but this view is naïve from both
195 business and technical standpoints.

196

197 *EDI* implementations encode substantial experience in *Business Processes*, and
198 companies with large investments in *EDI* integration will not abandon them without good
199 reason. *XML* enables more open, more flexible business transactions than *EDI*. *XML*
200 might enable more flexible and innovative "eMarketplace" business models than *EDI*.
201 But the challenges of designing *Messages* that meet *Business Process* requirements and
202 standardizing their semantics are independent of the syntax in which the *Messages* are
203 encoded.

204

205 The ebXML specifications provide a framework in which *EDI*'s substantial investments
206 in *Business Processes* can be preserved in an architecture that exploits *XML*'s new
207 technical capabilities.

208

209 Please consult the ebXML Requirements Specification, available at
210 <http://www.ebxml.org> for additional information on the underlying goals of ebXML.

211

212 **4.2 Caveats and Assumptions**

213 This specification is designed to provide a high level overview of ebXML, and as such,
214 does not provide the level of detail required to build *ebXML Applications*, components,
215 and related services. Please refer to each of the respective ebXML specifications to get
216 the level of detail.

217

218 **4.3 Design Conventions for ebXML Specifications**

219 In order to enforce a consistent capitalization and naming convention across all ebXML
220 specifications "Upper Camel Case" (*UCC*) and "Lower Camel Case" (*LCC*)

221 Capitalization styles SHALL be used. *UCC* style capitalizes the first character of each
222 word and compounds the name. *LCC* style capitalizes the first character of each word
223 except the first word.

224

225 1) ebXML DTD, XML Schema and *XML* instance documents SHALL have the effect of
226 producing ebXML *XML* instance documents such that:

227

- 228 • Element names SHALL be in *UCC* convention (example:
229 <UpperCamelCaseElement/>).
- 230 • Attribute names SHALL be in *LCC* convention (example:
231 <UpperCamelCaseElement lowerCamelCaseAttribute="Whatever"/>).

232

- 233 2) When *UML* and *Object Constrained Language (OCL)* are used to specify ebXML
234 artifacts Capitalization naming SHALL follow the following rules:
235
- 236 • Class, *Interface*, Association, Package, State, Use Case, Actor names SHALL use
237 UCC convention (examples: ClassificationNode, Versionable, Active,
238 InsertOrder, Buyer).
 - 239 • Attribute, Operation, Role, Stereotype, Instance, Event, Action names SHALL
240 use LCC convention (examples: name, notifySender, resident, orderArrived).
241
- 242 3) General rules for all names are:
- 243 • Acronyms SHOULD be avoided, but in cases where they are used, the
244 capitalization SHALL remain (example: XMLSignature).
 - 245 • Underscore (_), periods (.) and dashes (-) MUST NOT be used (don't use:
246 header.manifest, stock_quote_5, commercial-transaction, use HeaderManifest,
247 stockQuote5, CommercialTransaction instead).

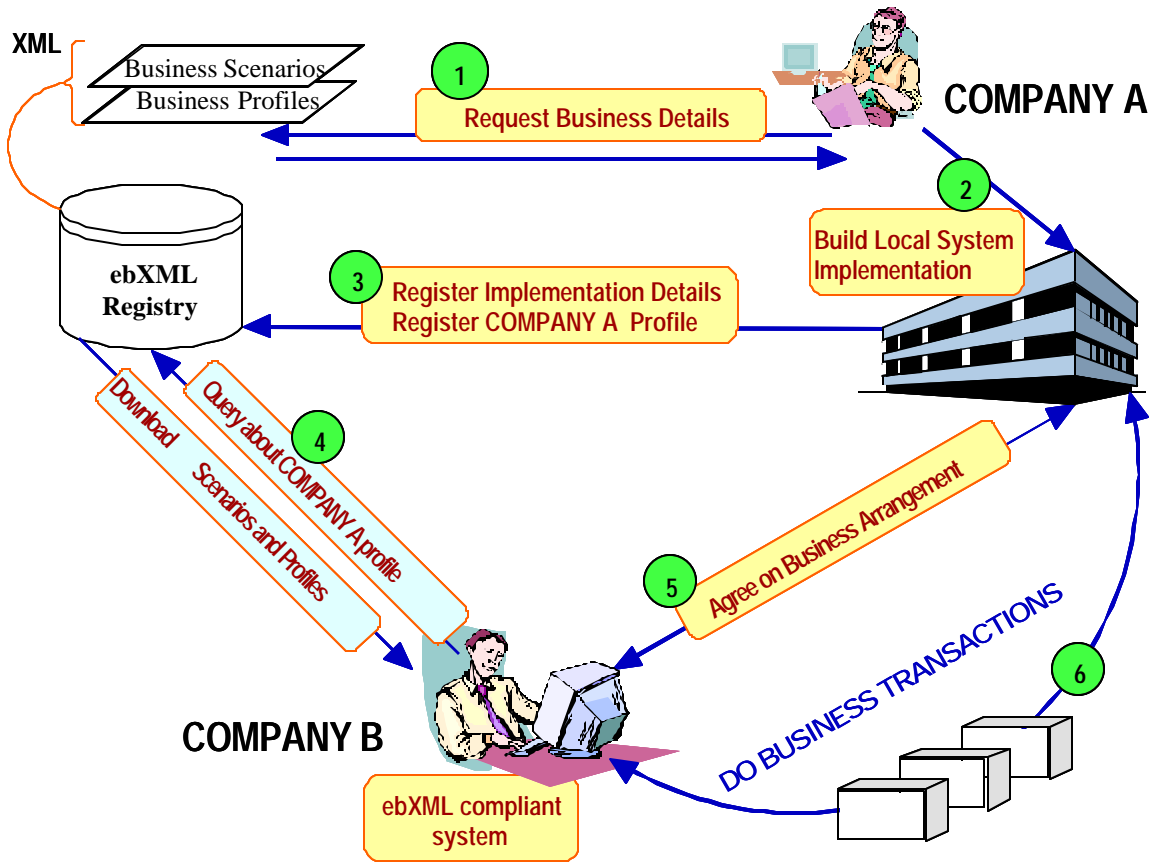
248 **5 ebXML System Overview**

249
250 Figure 1 below shows a high-level use case scenario for two *Trading Partners*, first
251 configuring and then engaging in a simple business transaction and interchange. This
252 model is provided as an example of the process and steps that may be required to
253 configure and deploy *ebXML Applications* and related architecture components. These
254 components can be implemented in an incremental manner. The ebXML specifications
255 are not limited to this simple model, provided here as quick introduction to the concepts.
256 Specific ebXML implementation examples are described in Appendix A.
257

258 The conceptual overview described below introduces the following concepts and
259 underlying architecture:
260

- 261 1. A standard mechanism for describing a *Business Process* and its associated
262 information model.
- 263 2. A mechanism for registering and storing *Business Process and Information Meta*
264 *Models* so they can be shared and reused.
- 265 3. Discovery of information about each participant including:
 - 266 • The *Business Processes* they support.
 - 267 • The *Business Service Interfaces* they offer in support of the *Business*
268 *Process*.
 - 269 • The *Business Messages* that are exchanged between their respective
270 *Business Service Interfaces*.
 - 271 • The technical configuration of the supported transport, security and
272 encoding protocols.
- 273 4. A mechanism for registering the aforementioned information so that it may be
274 discovered and retrieved.

- 275 5. A mechanism for describing the execution of a mutually agreed upon business
- 276 arrangement which can be derived from information provided by each participant
- 277 from item 3 above. (*Collaboration Protocol Agreement – CPA*)
- 278 6. A standardized business *Messaging Service* framework that enables interoperable,
- 279 secure and reliable exchange of *Messages* between *Trading Partners*.
- 280 7. A mechanism for configuration of the respective *Messaging Services* to engage in
- 281 the agreed upon *Business Process* in accordance with the constraints defined in
- 282 the business arrangement.



283
284
285
286

Figure 1 - a high level overview of the interaction of two companies conducting eBusiness using ebXML.

287 In Figure 1, Company A has become aware of an ebXML Registry that is accessible on
288 the Internet (Figure 1, step 1). Company A, after reviewing the contents of the ebXML
289 Registry, decides to build and deploy its own ebXML compliant application (Figure 1,
290 step 2). Custom software development is not a necessary prerequisite for ebXML
291 participation. ebXML compliant applications and components may also be commercially
292 available as shrink-wrapped solutions.

293

294 Company A then submits its own *Business Profile* information (including implementation
295 details and reference links) to the ebXML Registry (Figure 1, step 3). The business
296 profile submitted to the ebXML Registry describes the company’s ebXML capabilities
297 and constraints, as well as its supported business scenarios. These business scenarios are

298 XML versions of the *Business Processes* and associated information bundles (e.g. a sales
299 tax calculation) in which the company is able to engage. After receiving verification that
300 the format and usage of a business scenario is correct, an acknowledgment is sent to
301 Company A (Figure 1, step 3).

302

303 Company B discovers the business scenarios supported by Company A in the ebXML
304 *Registry* (Figure 1, step 4). Company B sends a request to Company A stating that they
305 would like to engage in a business scenario using ebXML (Figure 1, step 5). Company B
306 acquires an ebXML compliant shrink-wrapped application.

307

308 Before engaging in the scenario Company B submits a proposed business arrangement
309 directly to Company A's ebXML compliant software *Interface*. The proposed business
310 arrangement outlines the mutually agreed upon business scenarios and specific
311 agreements. The business arrangement also contains information pertaining to the
312 messaging requirements for transactions to take place, contingency plans, and security-
313 related requirements (Figure 1, step 5). Company A then accepts the business agreement.
314 Company A and B are now ready to engage in *eBusiness* using ebXML (Figure 1, step 6).

315 **6 ebXML Recommended Modeling Methodology**

316

317 *Business Process and Information Modeling* is not mandatory. However, if implementers
318 and users select to model *Business Processes and Information*, then they SHALL use the
319 *UN/CEFACT Modeling Methodology (UMM)* that utilizes *UML*.

320

321 **6.1 Overview**

322

323 While business practices from one organization to another are highly variable, most
324 activities can be decomposed into *Business Processes* which are more generic to a
325 specific type of business. This analysis through the modeling process will identify
326 *Business Process and Information Meta Models* that are likely candidates for
327 standardization. The ebXML approach looks for standard reusable components from
328 which to construct interoperable and components.

329

330 The *UN/CEFACT Modeling Methodology (UMM)* uses the following two views to
331 describe the relevant aspects of *eBusiness* transactions. This model is based upon the
332 Open-edi Reference Model, ISO/IEC 14662.

333

334

335

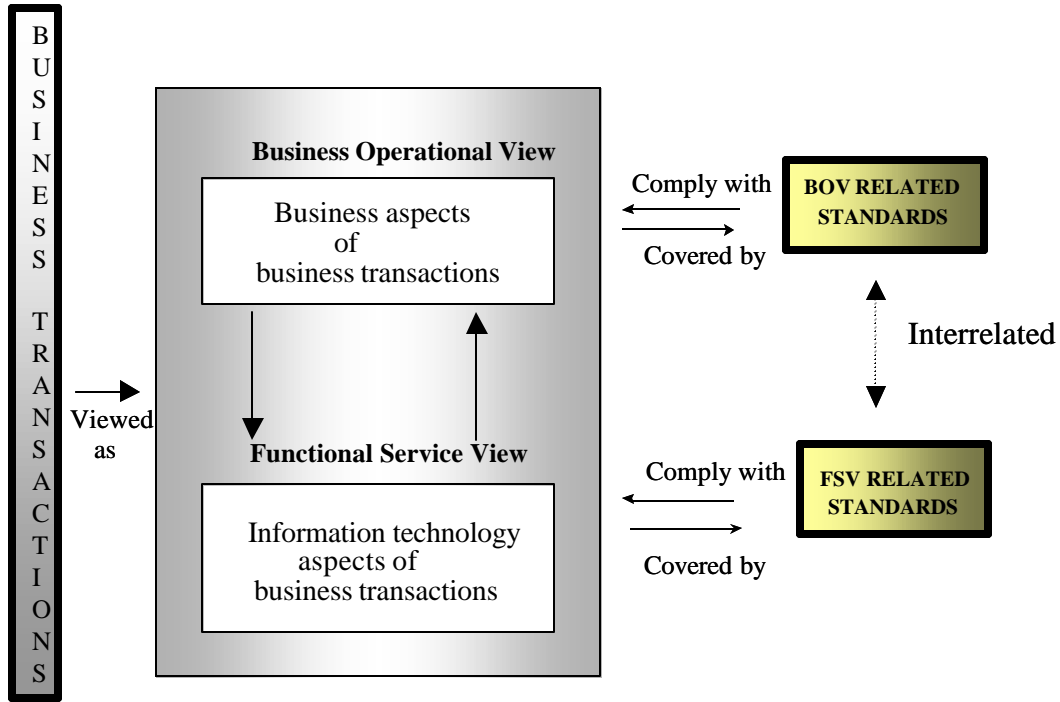


Figure 2 - ebXML Recommended Modeling Methodology

336
337

338

339 The *UN/CEFACT Modeling Methodology (UMM)* is broken down into the *Business*
 340 *Operational View (BOV)* and the supporting *Functional Service View (FSV)* described
 341 above. The assumption for ebXML is that the *FSV* serves as a reference model that *MAY*
 342 be used by commercial software vendors to help guide them during the development
 343 process. The underlying goal of the *UN/CEFACT Modeling Methodology (UMM)* is to
 344 provide a clear distinction between the operational and functional views, so as to ensure
 345 the maximum level of system interoperability and backwards compatibility with legacy
 346 systems (when applicable). As such, the resultant *BOV*-related standards provide the
 347 *UN/CEFACT Modeling Methodology (UMM)* for constructing *Business Process and*
 348 *Information Meta Models* for ebXML compliant applications and components.

349

350 The *BOV* addresses:

- 351 a) The semantics of business data in transactions and associated data interchanges
- 352 b) The architecture for business transactions, including:
 - 353 • operational conventions;
 - 354 • agreements and arrangements;
 - 355 • mutual obligations and requirements.

356

357 These specifically apply to the business needs of ebXML *Trading Partners*.

358

359 The *FSV* addresses the supporting services meeting the mechanistic needs of ebXML. It
 360 focuses on the information technology aspects of:

- 361 • Functional capabilities;
- 362 • *Business Service Interfaces*;

- Protocols and *Messaging Services*.

364

365 This includes, but is not limited to:

- Capabilities for implementation, discovery, deployment and run time scenarios;
- User *Interfaces*;
- Data transfer infrastructure *Interfaces*;
- *Protocols* for enabling interoperability of XML vocabulary deployments from different organizations.

370

371

372

6.2 ebXML Business Operational View

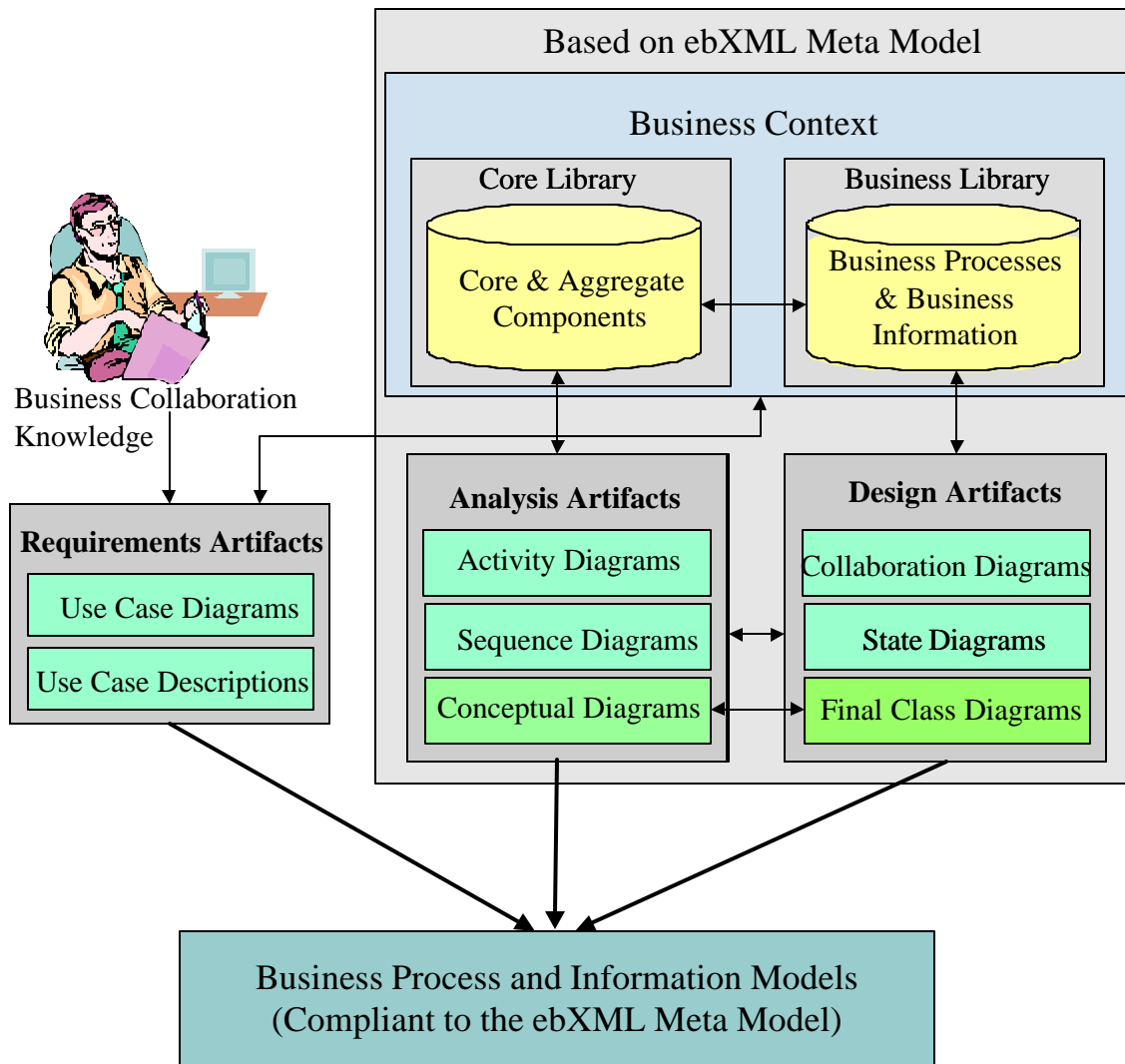
373

374

The modeling techniques described in this section are not mandatory requirements for participation in ebXML compliant business transactions.

375

376



377

378

379

Figure 3 – detailed representation of the Business Operational View

380

381 In Figure 3 above, *Business Collaboration Knowledge* is captured in a *Core Library*. The
382 *Core Library* contains data and process definitions, including relationships and cross-
383 references, as expressed in business terminology that MAY be tied to an accepted
384 industry classification scheme or taxonomy. The *Core Library* is the bridge between the
385 specific business or industry language and the knowledge expressed by the models in a
386 more generalized context neutral language.

387

388 The first phase defines the requirements artifacts that describe the problem using *Use*
389 *Case Diagrams and Descriptions*. If *Core Library* entries are available from an ebXML
390 compliant *Registry* they will be utilized, otherwise new *Core Library* entries will be
391 created and registered in an ebXML compliant *Registry*.

392

393 The second phase (analysis) will create activity and sequence diagrams (as defined in the
394 *UN/CEFACT Modeling Methodology* specification) describing the *Business Processes*.
395 *Class Diagrams* will capture the associated information parcels (business documents).
396 The analysis phase reflects the business knowledge contained in the *Core Library*. No
397 effort is made to force the application of object-oriented principles. The class diagram is
398 a free structured data diagram. Common *Business Processes* in the Business Library
399 MAY be referenced during the process of creating analysis and design artifacts.

400

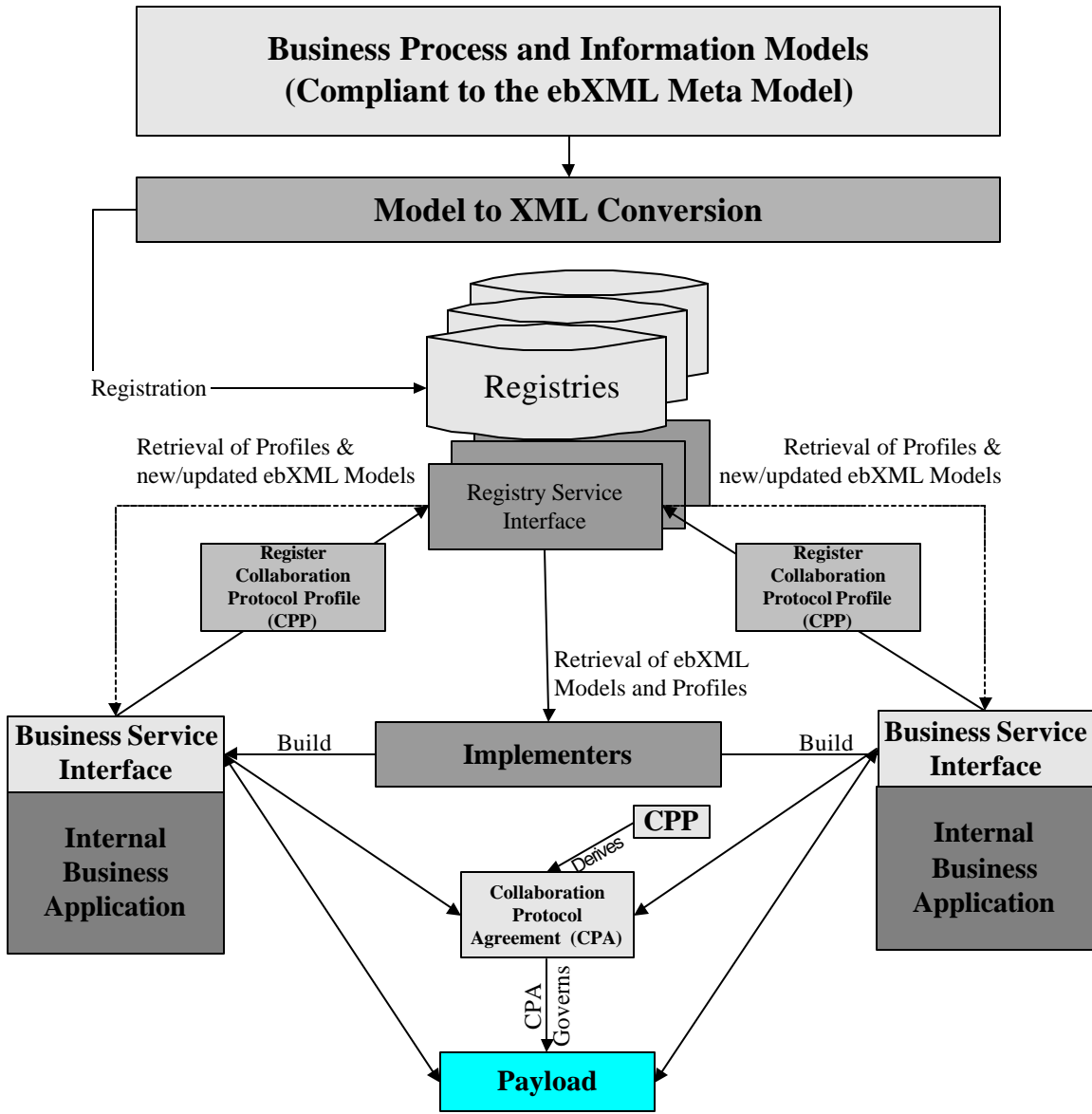
401 The design phase is the last step of standardization, which MAY be accomplished by
402 applying object-oriented principles based on the *UN/CEFACT Modeling Methodology*. In
403 addition to generating collaboration diagrams, a state diagram MAY also be created. The
404 class view diagram from the analysis phase will undergo harmonization to align it with
405 other models in the same industry and across others.

406

407 In ebXML, interoperability is achieved by applying *Business Information Objects* across
408 all class models. *Business Processes* are created by applying the *UN/CEFACT Modeling*
409 *Methodology (UMM)* which utilizes a common set of *Business Information Objects* and
410 *Core Components*.

411

412 **6.3 ebXML Functional Service View**
 413



414 **Figure 4 - ebXML Functional Service View**
 415
 416
 417

418 As illustrated in Figure 4 above, the ebXML *Registry Service* serves as the storage
 419 facility for the *Business Process and Information Models*, the *XML*-based representations
 420 of those models, *Core Components*, and *Collaboration Protocol Profiles*. The *Business*
 421 *Process and Information Meta Models* MAY be stored in modeling syntax, however they
 422 MAY be also stored as *XML* syntax in the *Registry*. This *XML*-based business
 423 information SHALL be expressed in a manner that allows discovery down to the atomic
 424 data level via a consistent methodology.
 425

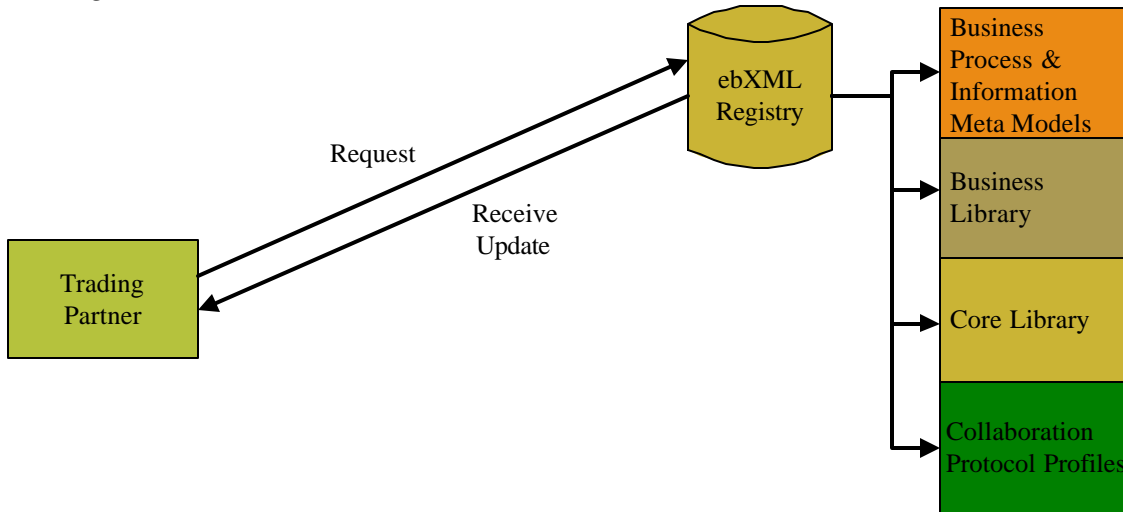
426 The underlying ebXML Architecture is distributed in such a manner to minimize the
 427 potential for a single point of failure within the ebXML infrastructure. This specifically
 428 refers to *Registry Services* (see Registry Functionality, Section 8.4 for details of this
 429 architecture).

430 **7 ebXML Functional Phases**

431
 432 **7.1 Implementation Phase**

433
 434 The implementation phase deals specifically with the procedures for creating an
 435 application of the ebXML infrastructure. A *Trading Partner* wishing to engage in an
 436 ebXML compliant transaction SHOULD first acquire copies of the ebXML
 437 Specifications. The *Trading Partner* studies these specifications and subsequently
 438 downloads the *Core Library* and the *Business Library*. The *Trading Partner* MAY also
 439 request other *Trading Partners' Business Process* information (stored in their business
 440 profile) for analysis and review. Alternatively, the *Trading Partner* MAY implement
 441 ebXML by utilizing 3rd party applications. The *Trading Partner* can also submit its own
 442 *Business Process* information to an ebXML compliant *Registry Service*.

443
 444 Figure 5 below, illustrates a basic interaction between an ebXML *Registry Service* and a
 445 *Trading Partner*.



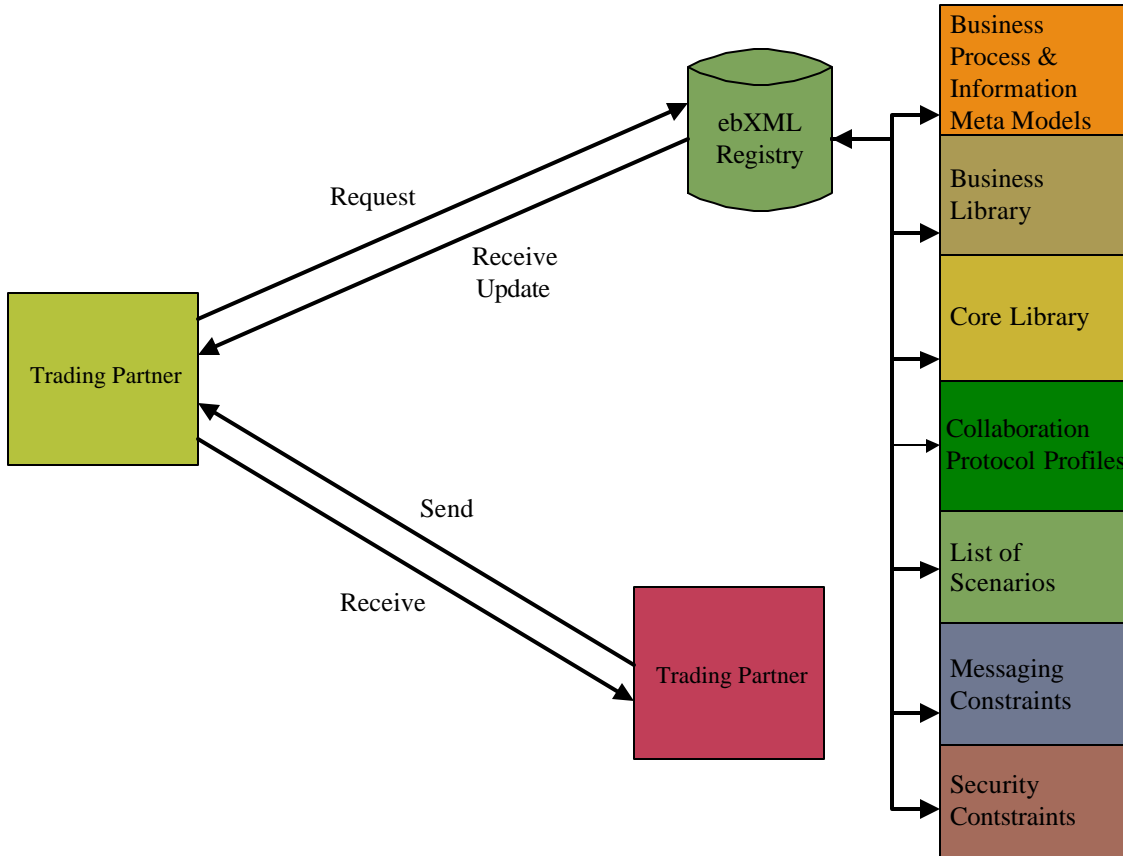
446
 447
 448
 449

Figure 5 - Functional Service View: Implementation Phase

450 **7.2 Discovery and Retrieval Phase**

451
 452 The Discovery and Retrieval Phase covers all aspects of the discovery of ebXML related
 453 resources. A *Trading Partner* who has implemented an ebXML *Business Service*
 454 *Interface* can now begin the process of discovery and retrieval (Figure 6 below). One
 455 possible discovery method may be to request the *Collaboration Protocol Profile* of
 456 another *Trading Partner*. Requests for updates to *Core Libraries*, *Business Libraries* and
 457 updated or new *Business Process and Information Meta Models* SHOULD be supported

458 by an ebXML *Business Service Interface*. This is the phase where *Trading Partners*
459 discover the meaning of business information being requested by other *Trading Partners*.
460



461
462
463

Figure 6 - Functional Service View: Discovery and Retrieval Phase

464

465 7.3 Run Time Phase

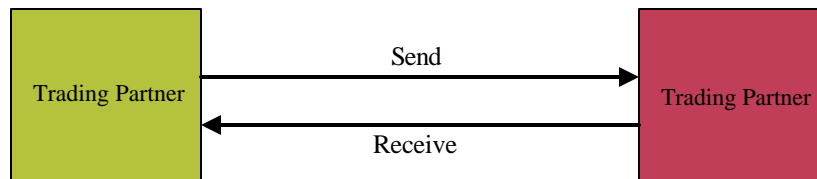
466

467 The run time phase covers the execution of an ebXML scenario with the actual associated
468 ebXML transactions. In the Run Time Phase, ebXML *Messages* are being exchanged
469 between *Trading Partners* utilizing the ebXML *Messaging Service*.

470

471 For example, an ebXML *CPA* is a choreographed set of business *Message* exchanges
472 linked together by a well-defined choreography using the ebXML *Messaging Service*.

473



474
475
476

Figure 7 - Functional Service View: Run Time Phase

477
478 [NOTE: There is no run time access to the *Registry*. If it becomes necessary to make calls
479 to the *Registry* during the run time, this SHOULD be considered as a reversion to the
480 Discovery and Retrieval Phase.]
481

482 **8 ebXML Infrastructure**

483

484 **8.1 Trading Partner Information [CPP and CPA's]**

485

486 **8.1.1 Introduction**

487 To facilitate the process of conducting *eBusiness*, potential *Trading Partners* need a
488 mechanism to publish information about the *Business Processes* they support along with
489 specific technology implementation details about their capabilities for exchanging
490 business information. This is accomplished through the use of a *Collaboration Protocol*
491 *Profile (CPP)*. The *CPP* is a document which allows a *Trading Partner* to express their
492 supported *Business Processes* and *Business Service Interface* requirements in a manner
493 where they can be universally understood by other ebXML compliant *Trading Partners*.
494

495 A special business agreement called a *CPA* is derived from the intersection of two or
496 more *CPP's*. The *CPA* serves as a formal handshake between two or more *Trading*
497 *Partners* wishing to conduct business transactions using ebXML.
498

499 **8.1.2 CPP Formal Functionality**

500 The *CPP* describes the specific capabilities that a *Trading Partner* supports as well as the
501 *Service Interface* requirements that need to be met in order to exchange business
502 documents with that *Trading Partner*. The *CPP* contains essential information about the
503 *Trading Partner* including, but not limited to: contact information, industry classification,
504 supported *Business Processes*, *Interface* requirements and *Messaging Service*
505 requirements. *CPP's* MAY also contain security and other implementation specific
506 details. Each ebXML compliant *Trading Partner* SHOULD register their *CPP(s)* in an
507 ebXML compliant *Registry Service*, thus providing a discovery mechanism that allows
508 *Trading Partners* to (1) find one another, (2) discover the *Business Process* that other
509 *Trading Partners* support.
510

511 The *CPP* definition SHALL provide for unambiguous selection of choices in all instances
512 where there may be multiple selections (e.g. HTTP or SMTP transport).
513

514 **8.1.3 CPA Formal Functionality**

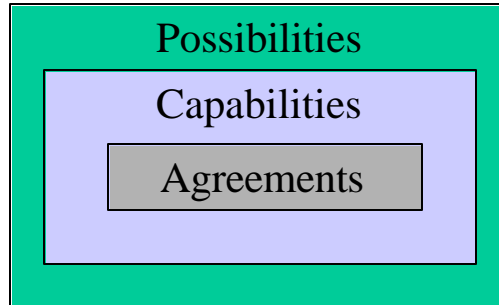
515 A *Collaboration Protocol Agreement (CPA)* is a document that represents the
516 intersection of two *CPP's* and is mutually agreed upon by both *Trading Partners* who
517 wish to conduct *eBusiness* using ebXML.
518

519 A *CPA* describes: (1) the *Messaging Service* and (2) the *Business Process* requirements
520 that are agreed upon by two or more *Trading Partners*. Conceptually, ebXML supports a
521 three level view of narrowing subsets to arrive at *CPA's* for transacting *eBusiness*. The

522 outer-most scope relates to all of the capabilities that a *Trading Partner* can support, with
523 a subset of what a *Trading Partner* “will” actually support.

524

525 A *CPA* contains the *Messaging Service Interface* requirements as well as the
526 implementation details pertaining to the mutually agreed upon *Business Processes* that
527 both *Trading Partners* agree to use to conduct *eBusiness*. *Trading Partners* may decide to
528 register their *CPA*’s in an ebXML compliant *Registry Service*, but this is not a mandatory
529 part of the *CPA* creation process.



530

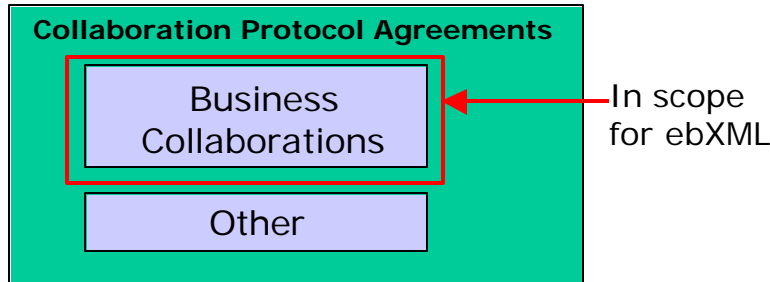
531

532 *Figure 8 - Three level view of CPA's*

533

534 *Business Collaborations* are the first order of support that can be claimed by ebXML
535 *Trading Partners*. This “claiming of support” for specific *Business Collaborations* is
536 facilitated by a distinct profile defined specifically for publishing, or advertising in a
537 directory service, such as an ebXML *Registry* or other available service. Figure 9 below
538 outlines the scope for *Collaboration Protocol Agreements* within ebXML.

539



540

541

542 *Figure 9 - Scope for CPA's*

543

544 The *CPA-CPP* specification includes a non-normative appendix that discusses *CPA*
545 composition and negotiation and includes advice as to composition and negotiation
546 procedures.

547

548 **8.1.4 CPP Interface s**

549

550 **Interface to Business Processes**

551 A *CPP* SHALL be capable of referencing one or more *Business Processes* supported by
552 the *Trading Partner* owning the *CPP* instance. The *CPP* SHALL reference the Roles
553 within a *Business Process* that the user is capable of assuming. An example of a Role
554 could be the notion of a “Seller” and “Buyer” within a “Purchasing” *Business Process*.

555
556 The *CPP* SHALL be capable of being stored and retrieved from an ebXML *Registry*
557 *Mechanism*

558
559 A *CPP* SHOULD also describe binding details that are used to build an ebXML *Message*
560 *Header*.

561 **8.1.5 CPA Interfaces**

562
563 A *CPA* governs the *Business Service Interface* used by a *Trading Partner* to constrain the
564 *Business Service Interface* to a set of parameters agreed to by all *Trading Partners* who
565 will execute such an agreement.

566
567 *CPA*'s have *Interfaces* to *CPP*'s in that the *CPA* is derived through a process of mutual
568 negotiation narrowing the *Trading Partners* capabilities (*CPP*) into what the *Trading*
569 *Partner* “will” do (*CPA*).

570
571 A *CPA* must reference to a specific *Business Process* and the interaction requirements
572 needed to execute that *Business Process*.

573
574 A *CPA* MAY be stored in a *Registry* mechanism, hence an implied ability to be stored
575 and retrieved is present.

576 **8.1.6 Non-Normative Implementation Details [CPP and CPA's]**

577
578
579 A *CPA* is negotiated after the Discovery and Retrieval Phase and is essentially a snapshot
580 of the *Messaging Services* and *Business Process* related information that two or more
581 *Trading Partners* agree to use to exchange business information. If any parameters
582 contained within an accepted *CPA* change after the agreement has been executed, a new
583 *CPA* SHOULD be negotiated between *Trading Partners*.

584
585 In some circumstances there may be a need or desire to describe casual, informal or
586 implied *CPA*'s.

587
588 An eventual goal of ebXML is to facilitate fully automated *CPA* generation. In order to
589 meet this goal, a formal methodology SHOULD be specified for the *CPA* negotiation
590 process.

591
592

593 **8.2 Business Process and Information Modeling**

594

595 **8.2.1 Introduction**

596 The ebXML *Business Process and Information Meta Model* is a mechanism that allows
 597 *Trading Partners* to capture the details for a specific business scenario using a consistent
 598 modeling methodology. A *Business Process* describes in detail how *Trading Partners*
 599 take on roles, relationships and responsibilities to facilitate interaction with other *Trading*
 600 *Partners* in shared collaborations. The interaction between roles takes place as a
 601 choreographed set of business transactions. Each business transaction is expressed as an
 602 exchange of electronic *Business Documents*. *Business Documents* MAY be composed
 603 from re-useable *Business Information Objects* (see “Relationships to Core Components”
 604 under 8.2.3 “Interfaces” below). At a lower level, *Business Processes* can be composed
 605 of re-useable *Core Processes*, and *Business Information Objects* can be composed of re-
 606 useable *Core Components*.

607

608 The ebXML *Business Process and Information Meta Model* supports requirements,
 609 analysis and design viewpoints that provide a set of semantics (vocabulary) for each
 610 viewpoint and forms the basis of specification of the artifacts that are required to
 611 facilitate *Business Process* and information integration and interoperability.

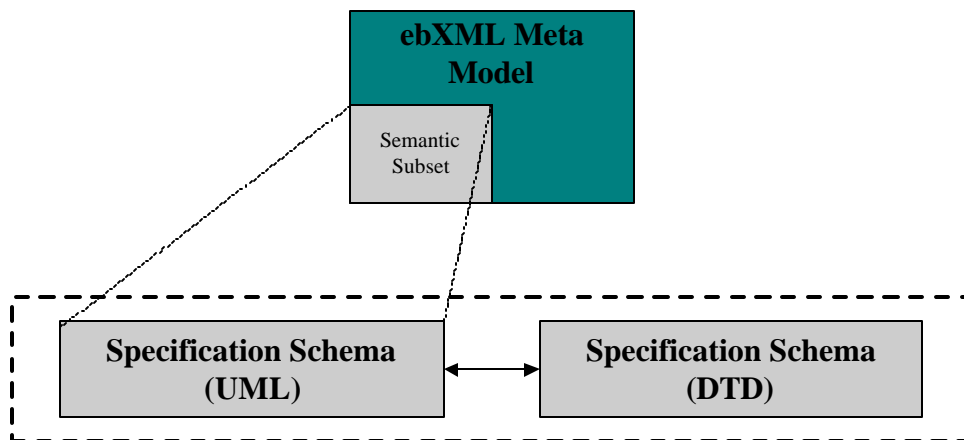
612

613 An additional view of the *Meta Model*, the *Specification Schema*, is also provided to
 614 support the direct specification of the set of elements required to configure a runtime
 615 system in order to execute a set of ebXML business transactions. By drawing out
 616 modeling elements from several of the other views, the *Specification Schema* forms a
 617 semantic subset of the ebXML *Business Process and Information Meta Model*. The
 618 *Specification Schema* is available in two stand-alone representations, a *UML* profile, and
 619 a *DTD*.

620

621 The relationship between the ebXML *Business Process and Information Meta Model* and
 622 the ebXML *Specification Schema* can be shown as follows:

623



624

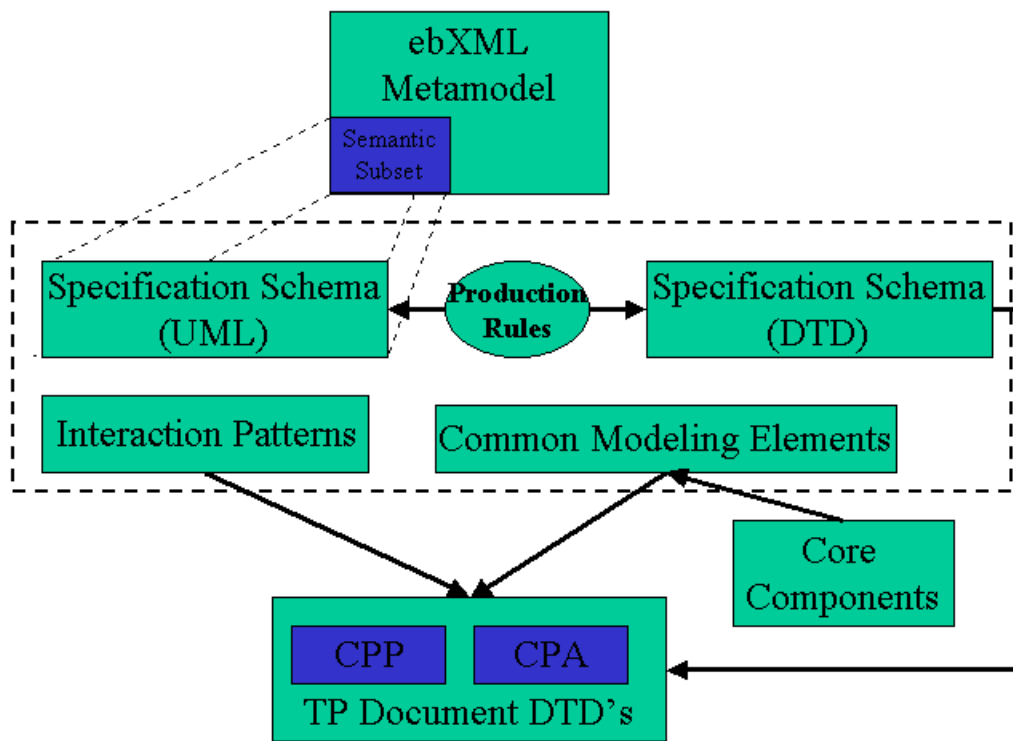
625

626

627

Figure 10 - ebXML Meta Model - Semantic Subset

628 The *Specification Schema* supports the specification of business transactions and the
 629 choreography of business transactions into *Business Collaborations*. Each *Business*
 630 *Transaction* can be implemented using one of many available standard patterns. These
 631 patterns determine the actual exchange of *Messages* and signals between *Trading*
 632 *Partners* to achieve the required electronic transaction. To help specify the patterns *the*
 633 *Specification Schema* is accompanied by a set of standard patterns, and a set of modeling
 634 elements common to those patterns. The full specification of a *Business Process* consists
 635 of a *Business Process and Information Meta Model* specified against the *Specification*
 636 *Schema* and an identification of the desired pattern(s). This information serves as the
 637 primary input for the formation of *Collaboration Protocol Profiles (CPP's)* and *CPA's*.
 638 This can be shown as follows:



639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649

Figure 11 - ebXML Meta Model

There are no formal requirements to mandate the use of a modeling language to compose new *Business Processes*, however, if a modeling language is used to develop *Business Processes*, it SHALL be the *Unified Modeling Language (UML)*. This mandate ensures that a single, consistent modeling methodology is used to create new *Business Processes*. One of the key benefits of using a single consistent modeling methodology is that it is possible to compare models to avoid duplication of existing *Business Processes*.

650 To further facilitate the creation of consistent *Business Processes* and information
651 models, ebXML will define a common set of *Business Processes* in parallel with a *Core*
652 *Library*. It is possible that users of the ebXML infrastructure may wish to extend this set
653 or use their own *Business Processes*.

654

655 **8.2.2 Formal Functionality**

656 The representation of a *Business Process* document instance SHALL be in a form that
657 will allow both humans and applications to read the information. This is necessary to
658 facilitate a gradual transition to full automation of business interactions.

659

660 The *Business Process* SHALL be storable and retrievable in a *Registry* mechanism.
661 *Business Processes* MAY be registered in an ebXML *Registry* in order to facilitate
662 discovery and retrieval.

663

664 To be understood by an application, a *Business Process* SHALL be expressible in *XML*
665 syntax. A *Business Process* MAY be constructed as an *Business Process and Information*
666 *Meta Model* or an *XML* representation of that model. *Business Processes* are capable of
667 expressing the following types of information:

- 668 • Choreography for the exchange of document instances. (e.g. the choreography of
669 necessary *Message* exchanges between two Trading Partners executing a
670 “Purchasing” ebXML transaction.)
- 671 • References to *Business Process and Information Meta Model* or *Business*
672 *Documents* (possibly *DTD*’s or *Schemas*) that add structure to business data.
- 673 • Definition of the roles for each participant in a *Business Process*.

674 A *Business Process*:

- 675 • Provides the contextual constraints for using *Core Components*
- 676 • Provides the framework for establishing *CPAs*
- 677 • Specifies the domain owner of a *Business Process*, along with relevant contact
678 information.

679 [NOTE: the above lists are not inclusive.]

680

681 **8.2.3 Interfaces**

682

683 **Relationship to CPP and CPA**

684 The *CPP* instance of a *Trading Partner* defines that partner’s functional and technical
685 capability to support zero, one, or more *Business Processes* and one or more roles in each
686 process.

687

688 The agreement between two *Trading Partners* defines the actual conditions under which
689 the two partners will conduct business transactions together. The *Interface* between the
690 *Business Process*, its *Information Meta Model*, and the *CPA* is the part of the *Business*
691 *Process* document. This MAY be instantiated as an *XML* document representing the
692 business transactional and collaboration layers of the *Business Process and Information*
693 *Meta Model*. The expression of the sequence of commercial transactions in *XML* is
694 shared between the *Business Process* and *Trading Partner Information* models.

695

696 **Relationship to Core Components**

697 A *Business Process* instance SHOULD specify the constraints for exchanging business
 698 data with other *Trading Partners*. The business information MAY be comprised of
 699 components of the ebXML *Core Library*. A *Business Process* document SHALL
 700 reference the *Core Components* directly or indirectly using a *XML* document that
 701 references the appropriate Business and Information Models and/or Business Documents
 702 (possibly DTD's or Schemas). The mechanism for interfacing with the *Core Components*
 703 and *Core Library* SHALL be by way of a unique identifier for each component.
 704

705 **Relationship to ebXML Messaging**

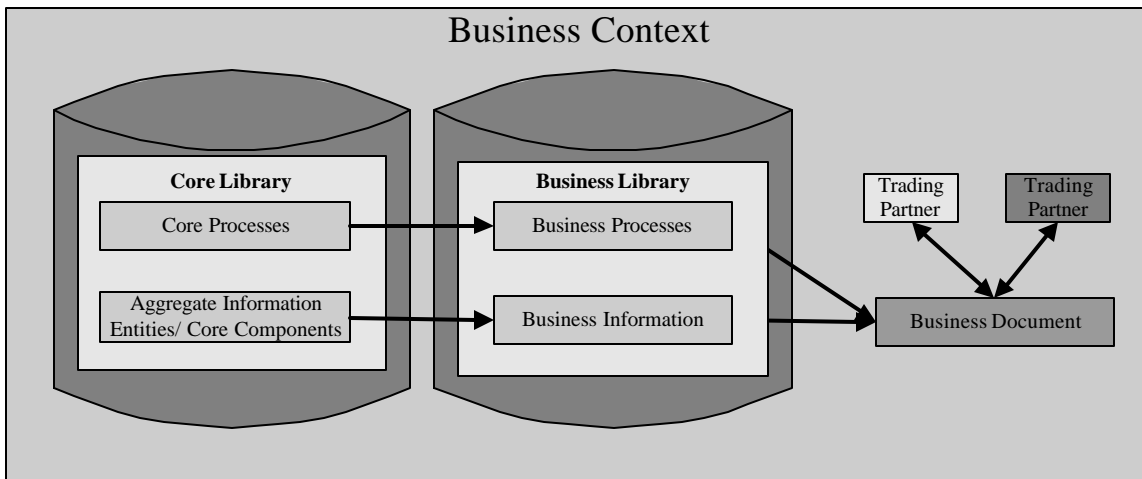
706 A *Business Process* instance SHALL be capable of being transported from a *Registry*
 707 *Service* to another *Registry Service* via an ebXML *Message*. It SHALL also be capable
 708 of being transported between a *Registry* and a users application via the ebXML
 709 *Messaging Service*.
 710

711 **Relationship to a Registry System**

712 A *Business Process* instance intended for use within the ebXML infrastructure SHALL
 713 be retrievable through a *Registry* query, and therefore, each *Business Process* SHALL
 714 contain a unique identifier.
 715

716 **8.2.4 Non-Normative Implementation Details**

717 The exact composition of *Business Information Objects* or a *Business Document* is
 718 guided by a set of contexts derived from the *Business Process*. The modeling layer of the
 719 architecture is highlighted in green in Figure 12 below.
 720



721
 722 **Figure 12 – ebXML Business Process and Information Modeling layer**
 723
 724

725 ebXML *Business Process and Information Meta Model* MAY be created following the
 726 recommended UN/CEFACT *Modeling Methodology (UMM)*, or MAY be arrived at in
 727 any other way, as long as they comply with the ebXML *Business Process and*
 728 *Information Meta Model*.
 729

730 **8.3 Core Components and Core Library Functionality**

731

732 **8.3.1 Introduction**

733

734 A *Core Component* captures information about a real world business concept, and the
735 relationships between that concept, other *Business Information Objects*, and a contextual
736 description that describes how a *Core* or *Aggregate Information Entity* may be used in a
737 particular ebXML *eBusiness* scenario.

738

739 A *Core Component* can be either an individual piece of business information, or a natural
740 “go-together” family of *Business Information Objects* that may be assembled into
741 *Aggregate Information Entities*.

742

743 The ebXML Core Components project team SHALL define an initial set of *Core*
744 *Components*. ebXML users may adopt and/or extend components from the ebXML *Core*
745 *Library*.

746

747 **8.3.2 Formal Functionality**

748

749 As a minimum set of requirements, *Core Components* SHALL facilitate the following
750 functionality:

751

752 *Core Components* SHALL be storable and retrievable using an ebXML *Registry*
753 *Mechanism*.

754

755 *Core Components* SHALL capture and hold a minimal set of information to satisfy
756 *eBusiness* needs.

757

758 *Core Components* SHALL be capable of being expressed in *XML* syntax.

759

760 A *Core Component* SHALL be capable of containing:

761

- 762 • Another *Core Component* in combination with one or more individual pieces of
763 *Business Information Objects*.

764

- 765 • Other *Core Components* in combination with zero or more individual pieces of
766 *Business Information Objects*.

767

768 A *Core Component* SHALL be able to be uniquely identified.

769

770 **8.3.3 Interfaces**

771

772 A *Core Component* MAY be referenced indirectly or directly from a *Business Document*
773 instance. The *Business Process* MAY specify a single or group of *Core Components* as
774 required or optional information as part of a *Business Document* instance.

775

776 A *Core Component* SHALL interface with a *Registry* mechanism by way of being
 777 storable and retrievable in such a mechanism.

778
 779 A *Core Component* MAY interface with an *XML Element* from another *XML* vocabulary
 780 by the fact it is bilaterally or unilaterally referenced as a semantic equivalent.

781
 782

783 **8.3.4 Non-Normative Implementation Details**

784

785 A *Core Component* MAY contain attribute(s) or be part of another *Core Component*, thus
 786 specifying the precise context or combination of contexts in which it is used.

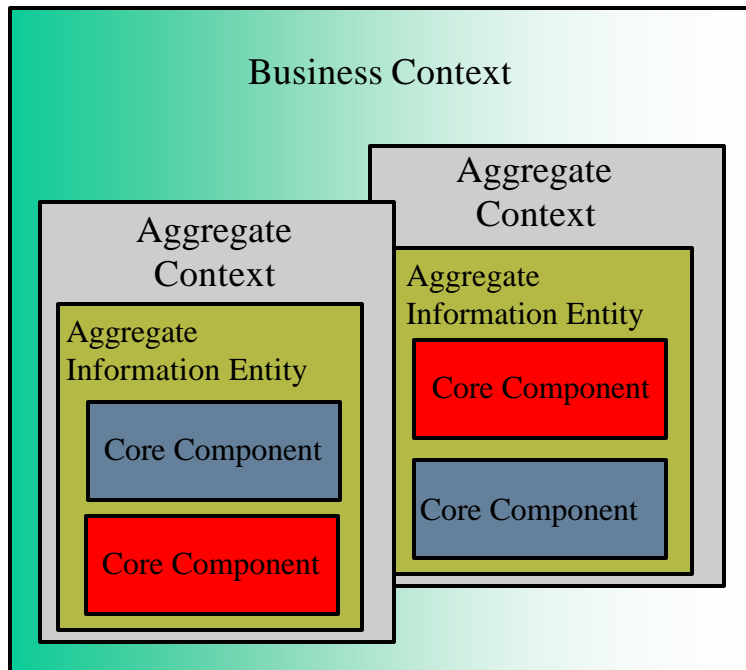
787

788 The process of aggregating *Core Components* for a specific business context, shall
 789 include a means to identify the placement of a *Core Component* within another *Core*
 790 *Component*. It MAY also be a combination of structural contexts to facilitate *Core*
 791 *Component* re-use at different layers within another *Core Component* or *Aggregate*
 792 *Information Entity*. This is referred to as *Business Context*.

793

794 Context MAY also be defined using the *Business Process and Information Meta Model*,
 795 which defines the instances of *Business Information Objects* in which the *Core*
 796 *Component* occurs.

797



798
 799

800 *Figure 13 - Business Context defined in terms of Aggregate Context, Aggregate Information Entities, and*
 801 *Core Components*

802

803 The pieces of *Business Information Objects*, or *Core Components*, within a generic *Core*
 804 *Component* may be either mandatory, or optional. A *Core Component* in a specific

805 context or combination of contexts (aggregate or business context) may alter the
 806 fundamental mandatory/optional cardinality.

807

808 **8.4 Registry Functionality**

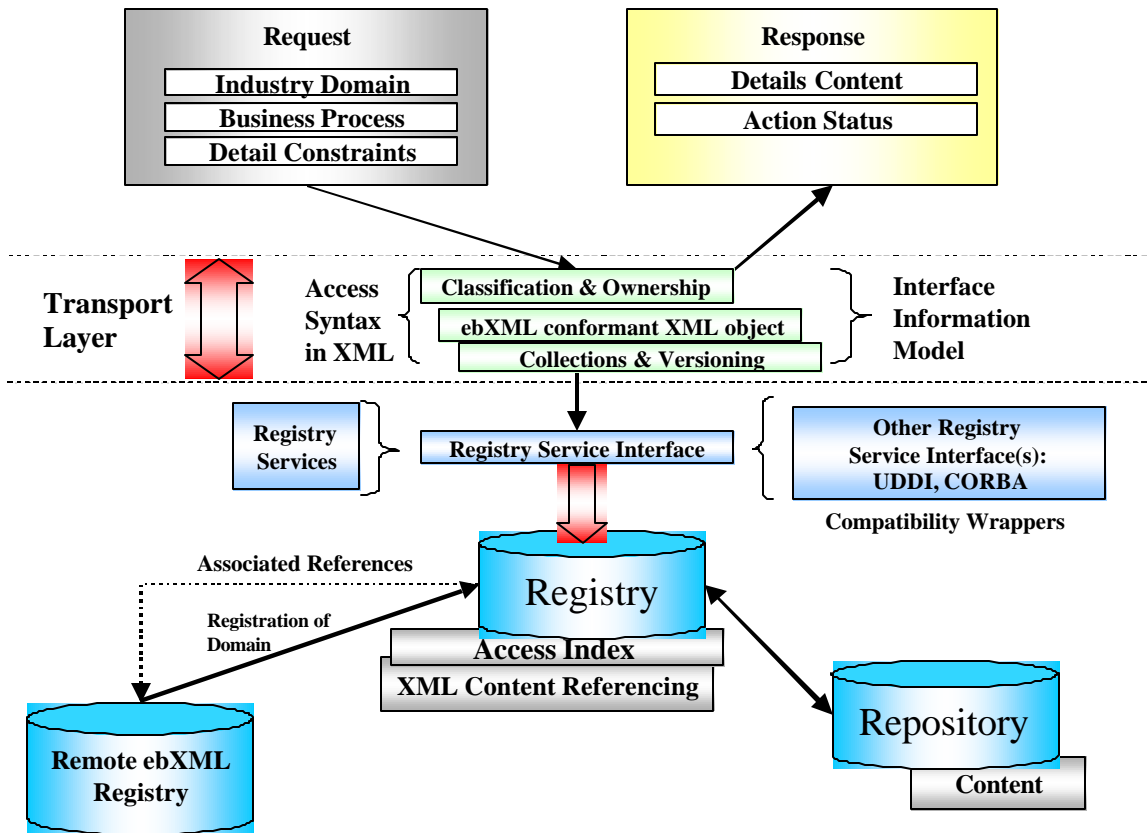
809

810 **8.4.1 Introduction**

811 An ebXML *Registry* provides a set of services that enable the sharing of information
 812 between *Trading Partners*. A *Registry* is a component that maintains an interface to
 813 metadata for a registered item. Access to an ebXML *Registry* is provided through
 814 *Interfaces* (APIs) exposed by *Registry Services*.

815

816



817

818

819

820

821 **8.4.2 Formal Functionality**

822 A *Registry* SHALL accommodate the storage of items expressed in syntax using multi-
 823 byte character sets.

824

825 Each *Registry Item*, at each level of granularity as defined by the *Submitting*
 826 *Organization*, MUST be uniquely identifiable. This is essential to facilitate application-
 827 to-Registry queries.

828

Figure 14 - Overall Registry Architecture.

829 A *Registry* SHALL return either zero or one positive matches in response to a contextual
830 query for a unique identifier. In such cases where two or more positive results are
831 displayed for such queries, an error message SHOULD be reported to the *Registry*
832 *Authority*.

833
834 A *Registry Item* SHALL be structured to allow information associations that identify,
835 name, describe it, give its administrative and access status, define its persistence and
836 mutability, classify it according to pre-defined classification schemes, declare its file
837 representation type, and identify the submitting and responsible organizations.

838
839 The *Registry Interface* serves as an application-to-registry access mechanism. Human-to-
840 registry interactions SHALL be built as a layer over a *Registry Interface* (e.g. a Web
841 browser) and not as a separate *Interface*.

842
843 The *Registry Interface* SHALL be designed to be independent of the underlying network
844 protocol stack (e.g. HTTP/SMTP over TCP/IP). Specific instructions on how to interact
845 with the *Registry Interface* MAY be contained in the payload of the ebXML *Message*.

846
847 The processes supported by the *Registry* MAY also include:

- 848 • A special *CPA* between the *Registry* and *Registry Clients*.
- 849 • A set of functional processes involving the *Registry* and *Registry Clients*.
- 850 • A set of *Business Messages* exchanged between a *Registry Client* and the *Registry*
851 as part of a specific *Business Process*.
- 852 • A set of primitive *Interface* mechanisms to support the *Business Messages* and
853 associated query and response mechanisms.
- 854 • A special *CPA* for orchestrating the interaction between ebXML compliant
855 Registries.
- 856 • A set of functional processes for *Registry-to-Registry* interactions.
- 857 • A set of error responses and conditions with remedial actions.

858
859 To facilitate the discovery process, browse and drill down queries MAY be used for
860 human interactions with a *Registry* (e.g. via a Web browser). A user SHOULD be able to
861 browse and traverse the content based on the available *Registry* classification schemes.

862
863 *Registry Services* exist to create, modify, and delete *Registry Items* and their metadata.

864
865 Appropriate security protocols MAY be deployed to offer authentication and protection
866 for the *Repository* when accessed by the *Registry*.

867
868 *Unique Identifiers (UIDs)* SHALL be assigned to all items within an ebXML *Registry*
869 *System*. *UID* keys are REQUIRED references for all ebXML content. *Universally Unique*
870 *Identifiers (UUIDs)* MAY be used to ensure that *Registry* entries are truly globally
871 unique, and thus when systems query a *Registry* for a *UUID*, one and only one result
872 SHALL be retrieved.

873

874 To facilitate semantic recognition of *Business Process and Information Meta Models*, the
875 *Registry Service* SHALL provide a mechanism for incorporating human readable
876 descriptions of *Registry* items. Existing *Business Process and Information Meta Models*
877 (e.g. RosettaNet PIPs) and *Core Components* SHALL be assigned *UID* keys when they
878 are registered in an ebXML compliant *Registry Service*. These *UID* keys MAY be
879 implemented in physical *XML* syntax in a variety of ways. These mechanisms MAY
880 include, but are not limited to:

881

- 882 • A pure explicit reference mechanism (example: URN:*UID* method),
- 883 • A referential method (example: URI:*UID* / namespace:*UID*),
- 884 • An object-based reference compatible with W3C Schema (*example*
885 URN:complextype name), and
- 886 • A datatype based reference (example: ISO 8601:2000 Date/Time/Number
887 datatyping and then legacy datatyping).

888

889 Components in ebXML MUST facilitate multilingual support. A *UID* reference is
890 particularly important here as it provides a language neutral reference mechanism. To
891 enable multilingual support, the ebXML specification SHALL be compliant with
892 Unicode and ISO/IEC 10646 for character set and UTF-8 or UTF-16 for character
893 encoding.

894

895 **8.4.3 Interfaces**

896

897 **ebXML Messaging :**

898 The query syntax used by the *Registry* access mechanisms is independent of the physical
899 implementation of the backend system.

900

901 The ebXML *Messaging Service* MAY serve as the transport mechanism for all
902 communication into and out of the *Registry*.

903

904 **Business Process:**

905 *Business Processes* are published and retrieved via ebXML *Registry Services*.

906

907 **Core Components:**

908 *Core Components* are published and retrieved via ebXML *Registry Services*.

909

910 **Any item with metadata:** *XML* elements provide standard metadata about the item being
911 managed through ebXML *Registry Services*. Since ebXML Registries are distributed
912 each *Registry* MAY interact with and cross-reference another ebXML *Registry*.

913

914 **8.4.4 Non-Normative Implementation Details**

915 The *Business Process and Information Meta Models* within a *Registry* MAY be stored
916 according to various classification schemes.

917

918 The existing ISO11179/3 work on *Registry* implementations MAY be used to provide a
919 model for the ebXML *Registry* implementation.

920

921 *Registry Items* and their metadata MAY also be addressable as *XML* based URI
 922 references using only HTTP for direct access.

923

924 Examples of extended *Registry Services* functionality may be deferred to a subsequent
 925 phase of the ebXML initiative. This includes, but is not limited to transformation
 926 services, workflow services, quality assurance services and extended security
 927 mechanisms.

928

929 A *Registry Service* MAY have multiple deployment models as long as the *Registry*
 930 *Interfaces* are ebXML compliant.

931

932 The *Business Process and Information Meta Model* for an ebXML *Registry Service* may
 933 be an extension of the existing OASIS Registry/Repository Technical Specification,
 934 specifically tailored for the storage and retrieval of business information, whereas the
 935 OASIS model is a superset designed for handling extended and generic information
 936 content.

937

938 **8.5 Messaging Service Functionality**

939

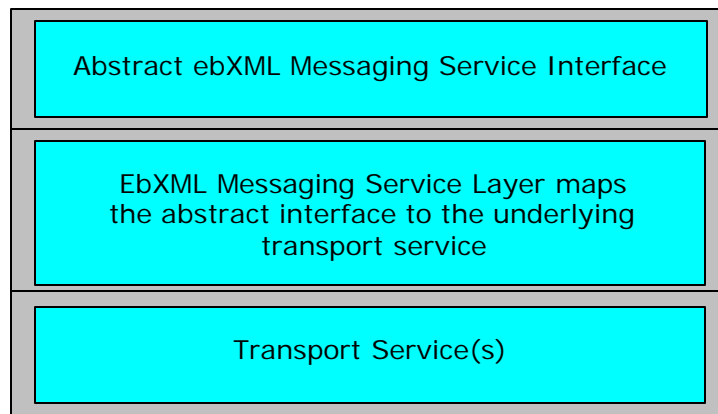
940 **8.5.1 Introduction**

941 The ebXML *Message Service* mechanism provides a standard way to exchange business
 942 *Messages* among ebXML *Trading Partners*. The ebXML *Messaging Service* provides a
 943 reliable means to exchange business *Messages* without relying on proprietary
 944 technologies and solutions. An ebXML *Message* contains structures for a *Message*
 945 *Header* (necessary for routing and delivery) and a *Payload* section.

946

947 The ebXML *Messaging Service* is conceptually broken down into three parts: (1) an
 948 abstract *Service Interface*, (2) functions provided by the *Messaging Service Layer*, and
 949 (3) the mapping to underlying transport service(s). The relation of the abstract *Interface*,
 950 *Messaging Service Layer*, and transport service(s) are shown in Figure 15 below.

951



952

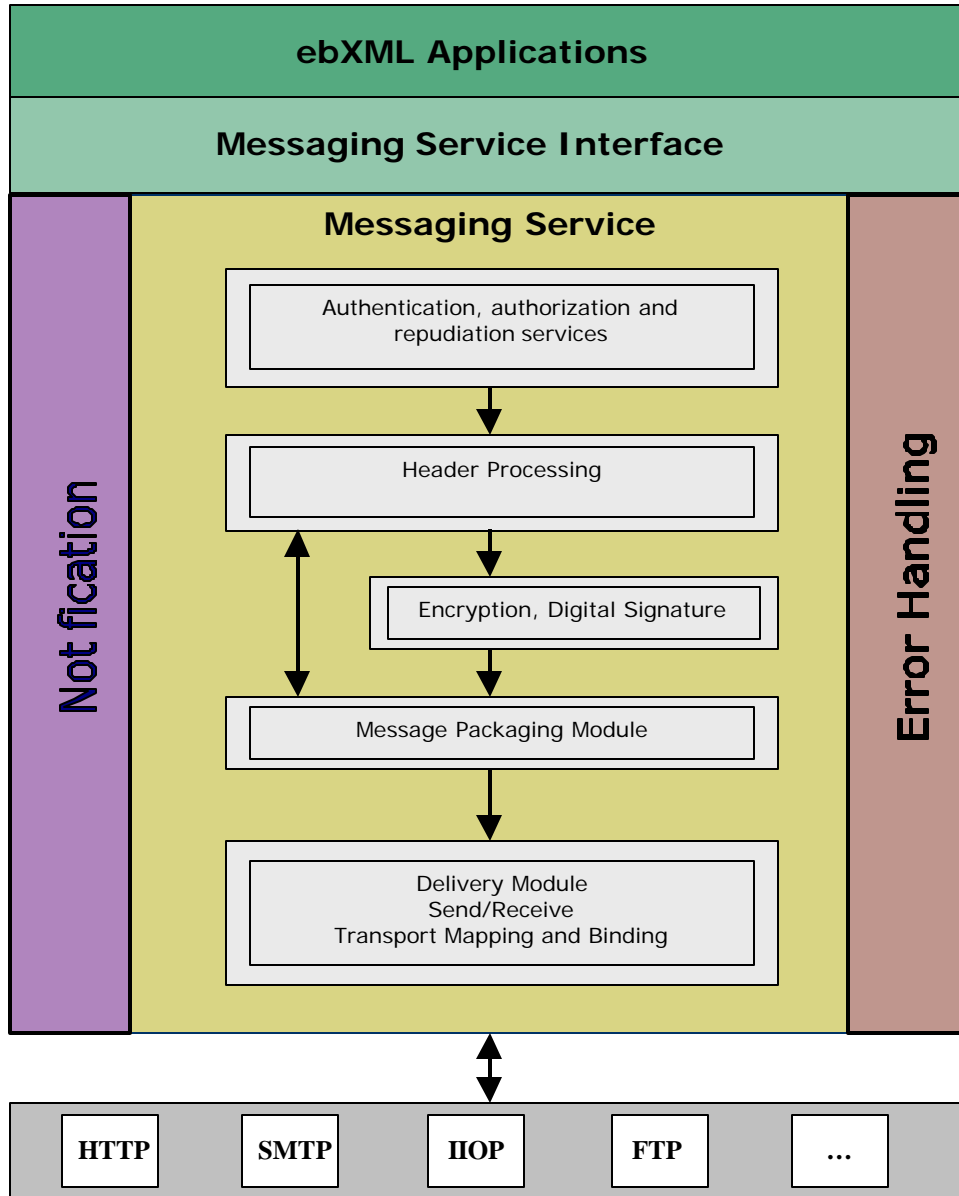
953

954

Figure 15 - ebXML Messaging Service

955
956
957
958
959
960
961

The following diagram depicts a logical arrangement of the functional modules that exist within the ebXML *Messaging Services* architecture. These modules are arranged in a manner to indicate their inter-relationships and dependencies. This architecture diagram illustrates the flexibility of the ebXML *Messaging Service*, reflecting the broad spectrum of services and functionality that may be implemented in an ebXML system.



962
963
964

Figure 16 - The Messaging Service Architecture

965 **8.5.2 Formal Functionality**

966 The ebXML *Messaging Service* provides a secure, consistent and reliable mechanism to
967 exchange ebXML *Messages* between users of the ebXML infrastructure over various
968 transport *Protocols* (possible examples include SMTP, HTTP/S, FTP, etc.).

969
970 The ebXML *Messaging Service* prescribes formats for all *Messages* between distributed
971 ebXML *Components* including *Registry* mechanisms and compliant user *Applications*.

972
973 The ebXML *Messaging Service* does not place any restrictions on the content of the
974 payload.

975
976 The ebXML *Messaging Service* supports simplex (one-way) and request/response (either
977 synchronous or asynchronous) *Message* exchanges.

978
979 The ebXML *Messaging Service* supports sequencing of payloads in instances where
980 multiple payloads or multiple *Messages* are exchanged between *Trading Partners*.

981
982 The ebXML *Messaging Service Layer* enforces the "rules of engagement" as defined by
983 two *Trading Partners* in a *Collaboration Protocol Agreement* (including, but not limited
984 to security and *Business Process* functions related to *Message* delivery). The
985 *Collaboration Protocol Agreement* defines the acceptable behavior by which each
986 *Trading Partner* agrees to abide. The definition of these ground rules can take many
987 forms including formal *Collaboration Protocol Agreements*, interactive agreements
988 established at the time a business transaction occurs (e.g. buying a book online), or other
989 forms of agreement. There are *Messaging Service Layer* functions that enforce these
990 ground rules. Any violation of the ground rules result in an error condition, which is
991 reported using the appropriate means.

992
993 The ebXML *Messaging Service* performs all security related functions including:

- 994 • Identification
- 995 • Authentication (verification of identity)
- 996 • Authorization (access controls)
- 997 • Privacy (encryption)
- 998 • Integrity (message signing)
- 999 • Non-repudiation
- 1000 • Logging

1001 **8.5.3 Interfaces**

1002 The ebXML *Messaging Service* provides ebXML with an abstract *Interface* whose
1003 functions, at an abstract level, include:

- 1004
- 1005
- 1006 • Send – send an ebXML *Message* – values for the parameters are derived from the
1007 ebXML *Message Headers*.
- 1008 • Receive – indicates willingness to receive an ebXML *Message*.
- 1009 • Notify – provides notification of expected and unexpected events.

- 1010 • Inquire – provides a method of querying the status of the particular ebXML
- 1011 *Message* interchange.

1012
 1013 The ebXML *Messaging Service* SHALL interface with internal systems including:

- 1014 • Routing of received *Messages* to internal systems
- 1015 • Error notification

1016
 1017 The ebXML *Messaging Service* SHALL help facilitate the *Interface* to an ebXML

1018 *Registry*.

1019
 1020 **8.5.4 Non-Normative Implementation Details**

1021
 1022 **ebXML Message Structure and Packaging**

1023
 1024 Figure 17 below illustrates the logical structure of an ebXML *Message*.

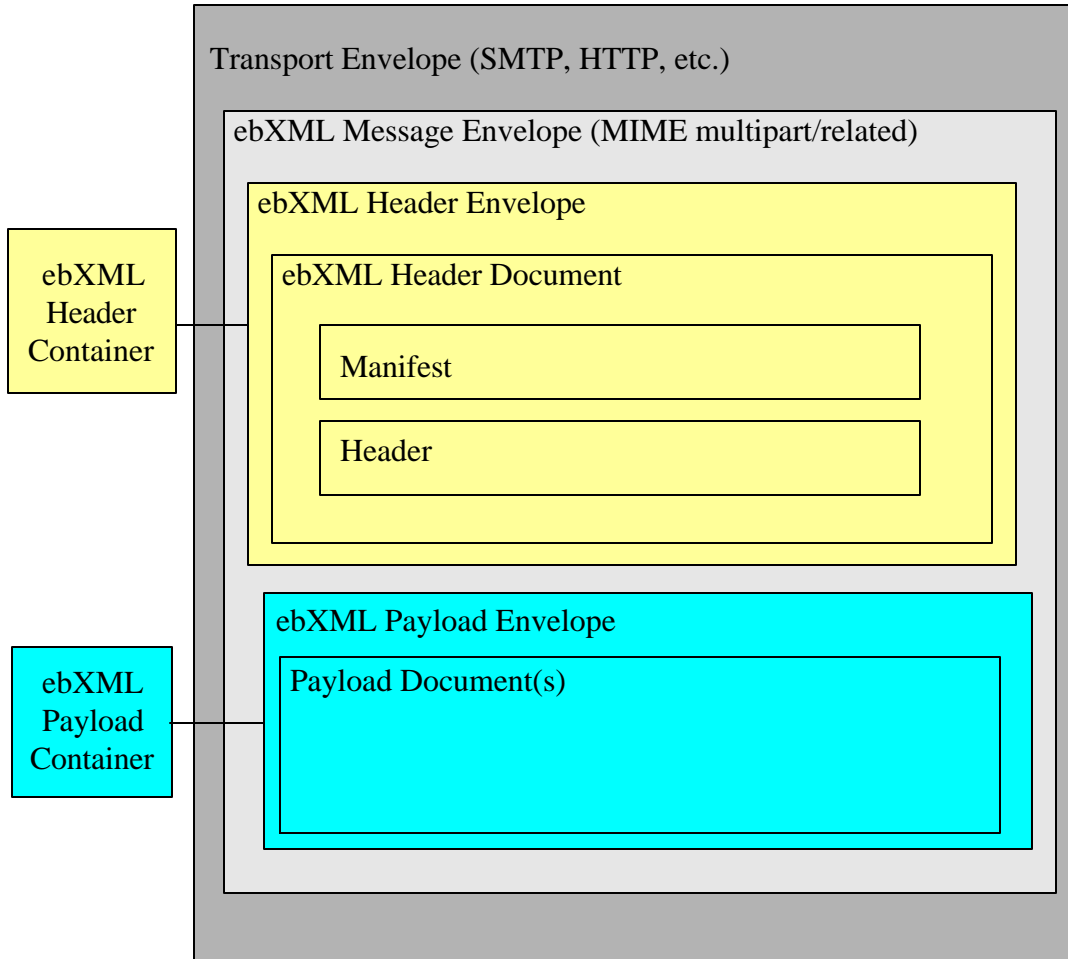


Figure 17 - ebXML Message Structure

1025
 1026
 1027
 1028

1029 An ebXML *Message* consists of an optional transport *Protocol* specific outer
1030 *Communication Protocol Envelope* and a *Protocol* independent ebXML *Message*
1031 *Envelope*. The ebXML *Message Envelope* is packaged using the MIME multipart/related
1032 content type. MIME is used as a packaging solution because of the diverse nature of
1033 information exchanged between *Partners* in *eBusiness* environments. For example, a
1034 complex *Business Transaction* between two or more *Trading Partners* might require a
1035 payload that contains an array of business documents (*XML* or other document formats),
1036 binary images, or other related Business Information.

1037 **9 Conformance**

1038

1039 **9.1 Introduction**

1040

1041 This clause specifies the general framework, concepts and criteria for *Conformance* to
1042 ebXML, including an overview of the conformance strategy for ebXML, guidance for
1043 addressing conformance in each ebXML technical specification, and the conformance
1044 clause specific to the Technical Architecture specification. Except for the Technical
1045 Architecture Specification, this clause does not define the conformance requirements for
1046 each of the ebXML technical specifications – the latter is the purview of the technical
1047 specifications.

1048

1049 The objectives of this section are to:

- 1050 a) Ensure a common understanding of conformance and what is required to claim
- 1051 conformance to this family of specifications;
- 1052 b) Ensure that conformance is consistently addressed in each of the component
- 1053 specifications;
- 1054 c) Promote interoperability and open interchange of *Business Processes* and
- 1055 *Messages*;
- 1056 d) Encourage the use of applicable conformance test suites as well as promote
- 1057 uniformity in the development of conformance test suites.

1058

1059 Conformance to ebXML is defined in terms of conformance to the ebXML infrastructure
1060 and conformance to each of the technical specifications for ebXML. The primary
1061 purpose of conformance to ebXML is to increase the probability of successful
1062 interoperability between implementations and the open interchange of *XML* business
1063 documents and *Messages*. Successful interoperability and open interchange is more
1064 likely to be achieved if implementations conform to the requirements in the ebXML
1065 specifications.

1066

1067 **9.2 Conformance to ebXML**

1068

1069 ebXML Conformance is defined as conformance to an ebXML system that is comprised
1070 of all the architectural components of the ebXML infrastructure and satisfies at least the
1071 minimum conformance requirements for each of the ebXML technical specifications,

1072 including the functional and *Interface* requirements in this Technical Architecture
1073 specification.

1074

1075 In the context of ebXML, an implementation is said to exhibit conformance if it complies
1076 with the requirements of each applicable ebXML technical specification. The
1077 conformance requirements are stated in the conformance clause of each technical
1078 specification of ebXML. The conformance clause specifies explicitly all the
1079 requirements that have to be satisfied to claim conformance to that specification. These
1080 requirements MAY be applied and grouped at varying levels within each specification.

1081

1082 **9.3 Conformance to the Technical Architecture Specification**

1083

1084 This section details the conformance requirements for claiming conformance to the
1085 Technical Architecture specification.

1086

1087 In order to conform to this specification, each ebXML technical specification:

1088

a) SHALL support all the functional and *Interface* requirements defined in this
1089 specification that are applicable to that technical specification;

1089

1090

b) SHALL NOT specify any requirements that would contradict or cause non-
1091 conformance to ebXML or any of its components;

1091

1092

c) MAY contain a conformance clause that adds requirements that are more specific
1093 and limited in scope than the requirements in this specification;

1093

1094

d) SHALL only contain requirements that are testable.

1095

1096 A conforming implementation SHALL satisfy the conformance requirements of the
1097 applicable parts of this specification and the appropriate technical specification(s).

1098

1099 **9.4 General Framework of Conformance Testing**

1100

1101 The objective of conformance testing is to determine whether an implementation being
1102 tested conforms to the requirements stated in the relative ebXML specification.

1103

1104

Conformance testing enables vendors to implement compatible and interoperable systems
1105 built on the ebXML foundations. ebXML *Implementations* and *Applications* SHOULD
1106 be tested to available test suites to verify their conformance to ebXML Specifications as
1107 soon as test suites are available.

1107

1108

Publicly available test suites from vendor neutral organizations such as OASIS and NIST
1109 SHOULD be used to verify the conformance of ebXML *Implementations*, *Applications*,
1110 and *Components* claiming conformance to ebXML. Open source reference
1111 implementations MAY be available to allow vendors to test their products for *Interface*
1112 compatibility, conformance, and interoperability.

1109

1110

1111

1112

1113

1114 **10.0 Security Considerations**

1115

1116 **10.1 Introduction**

1117 A comprehensive *Security Model* for ebXML will be expressed in a separate document.

1118 The *Security Model* will be applied to the entire *ebXML Infrastructure*, with the
1119 underlying goal of best meeting the needs of users of ebXML.

1120

1121 The Security Model will comply with security needs specified in the ebXML
1122 Requirements Document.

1123

1124 ***Disclaimer***

1125 The views and specification expressed in this document are those of the authors and are
1126 not necessarily those of their employers. The authors and their employers specifically
1127 disclaim responsibility for any problems arising from correct or incorrect implementation
1128 or use of this design.

1129 ***Copyright Statement***

1130 Copyright © ebXML 2001. All Rights Reserved.

1131

1132 This document and translations of it MAY be copied and furnished to others, and
1133 derivative works that comment on or otherwise explain it or assist in its implementation
1134 MAY be prepared, copied, published and distributed, in whole or in part, without
1135 restriction of any kind, provided that the above copyright notice and this paragraph are
1136 included on all such copies and derivative works. However, this document itself MAY
1137 not be modified in any way, such as by removing the copyright notice or references to
1138 ebXML, UN/CEFACT, or OASIS, except as required to translate it into languages other
1139 than English.

1140

1141 The limited permissions granted above are perpetual and will not be revoked by ebXML
1142 or its successors or assigns.

1143

1144 This document and the information contained herein is provided on an
1145 "AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR
1146 IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE
1147 USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1148 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1149 PARTICULAR PURPOSE.

1150

1150 **Appendix A: Example ebXML Business Scenarios**

1151 **Definition**

1152 This set of scenarios defines how ebXML compliant software could be used to implement
1153 popular, well-known *eBusiness* models.

1154 **Scope**

1155 These scenarios are oriented to properly position the ebXML specifications as a
1156 convenient mean for companies to properly run electronic business over the Internet
1157 using open standards. They bridge the specifications to real life uses.

1158 **Audience**

1159 Companies planning to use ebXML compliant software will benefit from these scenarios
1160 because they will show how these companies may be able to implement popular business
1161 scenarios onto the ebXML specifications.

1162 **List**

- 1163 a) Two *Trading Partners* set-up an agreement and run the associated electronic
1164 exchange.
- 1165 b) Three or more *Trading Partners* set-up a *Business Process* implementing a
1166 supply-chain and run the associated exchanges
- 1167 c) A Company sets up a Portal that defines a *Business Process* involving the use of
1168 external business services.
- 1169 d) Three or more *Trading Partners* conduct business using shared *Business*
1170 *Processes* and run the associated exchanges.

1171 **Scenario 1 : Two Trading Partners set-up an agreement and run** 1172 **the associated exchange**

1173 In this scenario:

- 1174 • Each *Trading Partner* defines its own Profile (*CPP*).
 - 1175 Each Profile references:
 - 1176 ○ One or more existing *Business Process* found in the ebXML *Registry*
 - 1177 ○ One of more *Message* Definitions. Each *Message* definition is built from
1178 reusable components (*Core Components*) found in the ebXML *Registry*
 - 1179 Each Profile (*CPP*) defines:
 - 1180 ○ The business transactions that the *Trading Partner* is able to engage into
 - 1181 ○ The Technical protocol (like HTTP, SMTP etc) and the technical properties
1182 (such as special encryption, validation, authentication) that the *Trading*
1183 *Partner* supports in the engagement
 - 1184 ○ The *Trading Partners* acknowledge each other profile and create a *CPA*.
 - 1185 • The *Trading Partners* implement the respective part of the Profile. This is done:
 - 1186 ○ Either by creating/configuring a *Business Service Interface*.
 - 1187 ○ Or properly upgrading the legacy software running at their side
- 1188 In both cases, this step is about :
- 1189 ○ Plugging the Legacy into the ebXML technical infrastructure as specified by
1190 the *Messaging Service*.
 - 1191 ○ Granting that the software is able to properly engage the stated conversations

- 1192 ○ Granting that the exchanges semantically conform to the agreed upon
- 1193 *Message Definitions*
- 1194 ○ Granting that the exchanges technically conform with the underlying ebXML
- 1195 *Messaging Service*.
- 1196 ● The *Trading Partners* start exchanging *Messages* and performing the agreed upon
- 1197 commercial transactions.
- 1198

1199 **Scenario 2: Three or more parties set-up a Business Process**
 1200 **implementing a supply-chain and run the associated exchanges**

1201 The simple case of a supply-chain involving two *Trading Partners* can be redefined in
 1202 terms of the Scenario 1.

1203
 1204 Here we are dealing with situations where more *Trading Partners* are involved. We
 1205 consider a supply chain of the following type:



1206
 1207
 1208
 1209
 1210
 1211
 1212 What fundamentally differs from Scenario 1 is that “*Trading Partner 2*” is engaged at the
 1213 same time with two different *Trading Partners*. The assumption is that the “state” of the
 1214 local portion of the *Business Process* is managed by each *Trading Partner*, i.e. that each
 1215 *Trading Partner* is fully responsible of the Business Transaction involving it (“*Trading*
 1216 *Partner 3*” only knows about “*Trading Partner 2*”, “*Trading Partner 2*” knows about
 1217 “*Trading Partner 3*” and “*Trading Partner 1*”, “*Trading Partner 1*” knows about
 1218 “*Trading Partner 2*”).

1219
 1220 In this scenario:

- 1221 ● Each *Trading Partner* defines its own Profile (*CPP*). Each Profile (*CPP*)
- 1222 references:
 - 1223 ○ One or more existing *Business Process* found in the ebXML *Registry*
 - 1224 ○ One of more *Message Definitions*. Each *Message* definition is built from
 - 1225 reusable components (*Core Components*) found in the ebXML *Registry*
- 1226 Each Profile (*CPP*) defines:
 - 1227 ○ The Commercial Transactions that the *Trading Partner* is able to engage into.
 - 1228 “*Trading Partner 2*” must be able to support at least 2 Commercial
 - 1229 Transactions.
 - 1230 ○ The Technical protocol (like HTTP, SMTP etc) and the technical properties
 - 1231 (such as special encryption, validation, authentication) that the *Trading*
 - 1232 *Partner* supports in the engagement. As to “*Trading Partner 2*”, the technical
 - 1233 requirements for the exchanges with “*Trading Partner 1*” and “*Trading*
 - 1234 *Partner 3*” may be different. In such case, “*Trading Partner 2*” must be able
 - 1235 to support different protocols and/or properties.

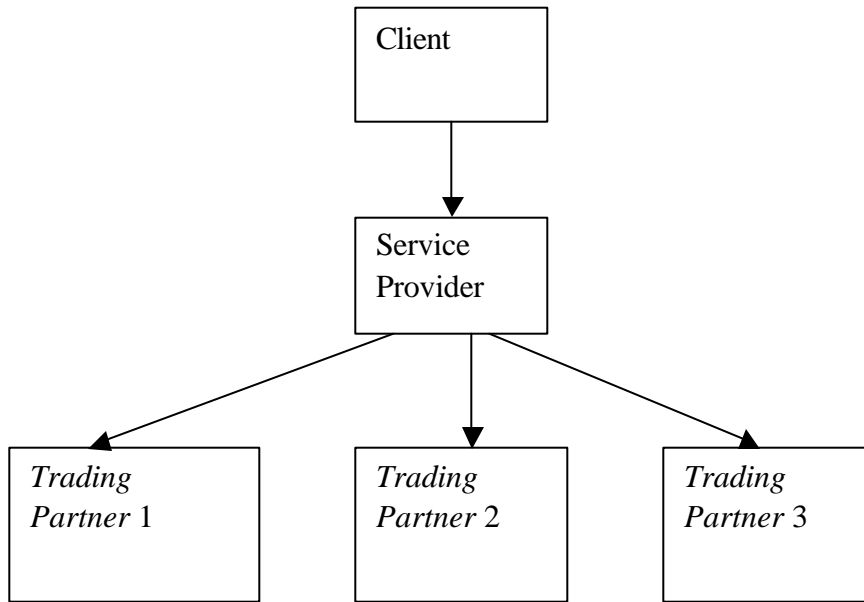
- 1236 ○ The *Trading Partners* acknowledge each other profile and create the relevant
1237 CPA. (at least 2 in this Scenario).
1238 ○ “*Trading Partner 2*” is engaged in 2 CPA’s
1239 • The *Trading Partners* implement the respective part of the Profile. This is done:
1240 ○ Either by creating/configuring a *Business Service Interface*.
1241 ○ Or properly upgrading the legacy software running at their side.
1242 In both cases, this step is about:
1243 ○ Plugging the Legacy into the ebXML technical infrastructure as specified by
1244 the *Messaging Service*
1245 ○ Granting that the software is able to properly engage the stated conversations
1246 ○ Granting that the exchanges semantically conform to the agreed upon ebXML
1247 *Message* definitions
1248 ○ Granting that the exchanges technically conform with the underlying ebXML
1249 *Messaging Service*.
1250 ○ “*Trading Partner 2*” may need to implement a complex *Business Service*
1251 *Interface* in order to be able to engage with different *Trading Partners*.
1252 • The *Trading Partners* start exchanging *Messages* and performing the agreed upon
1253 commercial transactions.
1254 ○ “*Trading Partner 3*” places an order at “*Trading Partner 2*”
1255 ○ “*Trading Partner 2*” (eventually) places an order with “*Trading Partner 1*”
1256 ○ “*Trading Partner 1*” fulfills the order
1257 ○ “*Trading Partner 2*” fulfill the order
1258

1259 **Scenario 3 : A Company sets up a Portal which defines a**
1260 ***Business Process involving the use of external business***
1261 ***services***

1262 This is the Scenario describing a Service Provider. A “client” asks the Service Provider
1263 for a Service. The Service Provider fulfills the request by properly managing the
1264 exchanges with other *Trading Partners* that provide information to build the final answer.
1265

1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286

In the simplest case, this Scenario could be modeled as follows :

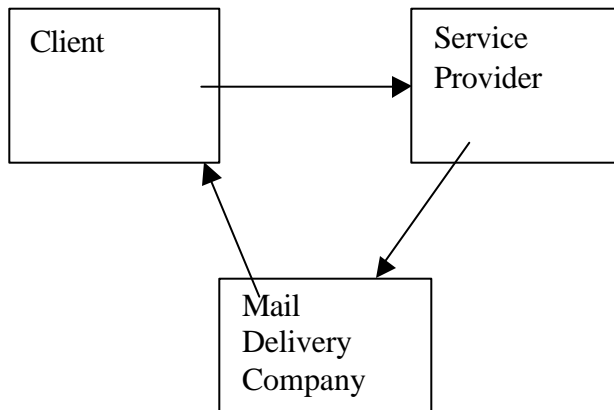


This is an evolution of Scenario 2. The Description of this scenario is omitted.

1287 **Scenario 4: Three or more Trading Partners conduct business**
1288 **using shared Business Processes and run the associated**
1289 **exchanges**

1290 This Scenario is about 3 or more *Trading Partners* having complex relationships. An
1291 example of this is the use of an external delivery service for delivering goods.

1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305



1306 In this Scenario, each *Trading Partner* is involved with more than one other *Trading*
1307 *Partner* but the relationship is not linear. The product or good that is ordered by the
1308 *Client* with a *Service Provider* is delivered by a 3rd party.

1309 In this scenario:

- 1310 • Each *Trading Partner* defines its own Profile (*CPP*). Each Profile (*CPP*)
1311 references:
- 1312 ○ One or more existing *Business Process* found in the ebXML *Registry*
 - 1313 ○ One of more *Registry* Definitions. Each *Registry* definition is built from
1314 reusable components (*Core Components*) found in the ebXML *Registry*
- 1315 Each Profile (*CPP*) defines:
- 1316 ○ The Commercial Transactions that the *Trading Partner* is able to engage into.
1317 In this case, each *Trading Partner* must be able to support at least 2
1318 Commercial Transactions.
 - 1319 ○ The Technical protocol (like HTTP, SMTP etc) and the technical properties
1320 (such as special encryption, validation, authentication) that the *Trading*
1321 *Partner* supports in the engagement.
1322 In case the technical infrastructure underlying the different exchanges differs,
1323 each *Trading Partner* must be able to support different protocols and/or
1324 properties. (an example is that the order is done through a Web Site and the
1325 delivery is under the form of an email).
 - 1326 ○ The *Trading Partners* acknowledge each other profile and create a *CPA*.
1327 Each *Trading Partner*, in this Scenario, must be able to negotiate at least 2
1328 Agreements.
- 1329 Each *Trading Partner* is engaged in 2 Agreements (*CPA*).
- 1330 • The *Trading Partners* implement the respective part of the Profile. This is done:
1331 ○ Either by creating/configuring a *Business Service Interface*.
1332 ○ Or properly upgrading the legacy software running at their side
- 1333 In both cases, this step is about:
- 1334 ○ Plugging the Legacy into the ebXML technical infrastructure as specified by
1335 the *Messaging Service*.
 - 1336 ○ Granting that the software is able to properly engage the stated conversations
 - 1337 ○ Granting that the exchanges semantically conform to the agreed upon
1338 *Message* definitions.
 - 1339 ○ Granting that the exchanges technical conform with the underlying ebXML
1340 *Messaging Service*.
 - 1341 ○ All *Trading Partners* may need to implement complex *Business Service*
1342 *Interfaces* to accommodate the differences in the *CPA*'s with different
1343 *Trading Partners*.
- 1344 • The *Trading Partners* start exchanging *Messages* and performing the agreed upon
1345 commercial transactions.
 - 1346 ○ The *Client* places an Order at the Service Provider.
 - 1347 ○ The Service Provider Acknowledges the Order with The *Client*.
 - 1348 ○ The Service Provider informs the Mail Delivery Service about a good to be
1349 delivered at the *Client*
 - 1350 ○ The Mail Delivery Service delivers the good at the *Client*
 - 1351 ○ The *Clients* notifies the Service Provider that the good is received.