



Creating A Single Global Electronic Market

1

2 **ebXML Technical Architecture Specification**

3 **ebXML Technical Architecture Project Team**

4

5 4 January 2001

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

7

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

12

13 ***This version:***

14 ebXML_TA_v1.0.doc

15

16 ***Latest version:***

17 ebXML_TA_v1.0.doc

18

19 ***Previous version:***

20 ebXML_TA_v0.95.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	Melanie McCarthy, General Motors
46	Bruce Peat, eProcessSolutions
47	John Petit, KPMG Consulting
48	Mark Heller, MITRE
49	Scott Hinkelman, IBM
50	Karsten Riemer, Sun Microsystems
51	Lynne Rosenthal, NIST
52	Nikola Stojanovic, Columbine JDS Systems
53	Jeff Sutor, Sun Microsystems
54	David RR Webber, XML Global Technologies
55	

55 **4 Introduction**

56
57 **4.1 Summary of Contents of Document**
58

59 **EBXML TECHNICAL ARCHITECTURE SPECIFICATION 1**

60 1 STATUS OF THIS DOCUMENT 1

61 2 EBXML TECHNICAL ARCHITECTURE PARTICIPANTS 1

62 4 INTRODUCTION 3

63 4.1 *Summary of Contents of Document*..... 3

64 4.2 *Audience and Scope* 4

65 4.3 *Related Documents* 5

66 4.4 *Normative References* 5

67 4.5 *Document Conventions* 5

68 5 DESIGN OBJECTIVES 6

69 5.1 *Problem Description & Goals for ebXML*..... 6

70 5.2 *Caveats and Assumptions* 6

71 6 EBXML SYSTEM OVERVIEW 6

72 7 EBXML ARCHITECTURE REFERENCE MODEL..... 9

73 7.1 *Overview* 9

74 7.2 *ebXML Business Operational View* 10

75 7.3 *ebXML Functional Service View*..... 13

76 8 EBXML FUNCTIONAL PHASES 14

77 8.1 *Overview* 14

78 8.1.1 *The Implementation Phase*..... 14

79 8.1.2 *The Discovery and Retrieval Phase* 14

80 8.1.3 *The Run Time Phase* 14

81 8.2 *Implementation Phase*..... 14

82 8.3 *Discovery and Retrieval Phase* 15

83 8.4 *Run Time Phase* 16

84 9 EBXML INFRASTRUCTURE 17

85 9.1 *Trading Partner Information [CPP and CPA's]* 17

86 9.1.1 *Introduction*..... 17

87 9.1.2 *CPP Formal Functionality*..... 17

88 9.1.3 *CPA Formal Functionality*..... 17

89 9.1.4 *CPP Interfaces*..... 18

90 9.1.5 *CPA Interfaces* 19

91 9.1.6 *Non-Normative Implementation Details [CPP and CPA's]* 19

92 9.2 *Business Process and Information Modeling* 19

93 9.2.1 *Introduction*..... 19

94 9.2.2 *Formal Functionality*..... 21

95 9.2.3 *Interfaces* 22

96 9.2.4 *Non-Normative Implementation Details*..... 23

97 9.3 *Core Components and Core Library Functionality*..... 23

98 9.3.1 Introduction..... 23

99 9.3.2 Formal Functionality..... 24

100 9.3.3 Interfaces 24

101 9.3.4 Non-Normative Implementation Details..... 24

102 9.4 *Registry Functionality*..... 25

103 9.4.1 Introduction..... 25

104 9.4.2 Formal Functionality..... 26

105 9.4.3 Interfaces 27

106 9.4.4 Non-Normative Implementation Details..... 28

107 9.5 *Messaging Service Functionality*..... 28

108 9.5.1 Introduction..... 28

109 9.5.2 Formal Functionality..... 30

110 9.5.3 Interfaces 31

111 9.5.4 Non-Normative Implementation Details..... 32

112 10 CONFORMANCE..... 33

113 10.1 *Introduction*..... 33

114 10.2 *Conformance to ebXML*..... 33

115 10.3 *Conformance to the Technical Architecture Specification* 34

116 10.4 *General Framework of Conformance Testing* 34

117 11.0 SECURITY CONSIDERATIONS..... 34

118 11.1 *Introduction*..... 34

119 APPENDIX A: EXAMPLE EBXML BUSINESS SCENARIOS 35

120 *Scenario 1* 35

121 *Two Trading Partners set-up an agreement and run the associated exchange*..... 35

122 *Scenario 2:* 36

123 *Three or more Trading Partners set-up a Business Process implementing a*

124 *supply-chain eBusiness scenario.*..... 36

125 *Scenario 3* 38

126 *A Company sets up a Portal which defines a Business Process involving the use*

127 *of external business services*..... 38

128 *Scenario 4* 38

129 *Three or more Trading Partners engage in eBusiness using Business Processes*

130 *that were created by each respective Trading Partner and run the associated*

131 *business exchanges.* 38

132 DISCLAIMER..... 40

133 COPYRIGHT STATEMENT..... 40

134

135 **4.2 Audience and Scope**

136

137 This document is intended primarily for the *ebXML Project Teams* to help guide their

138 work. Secondary audiences MAY include software implementers, international standards

139 bodies, and other industry organizations.

140

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

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

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

145

146 **4.3 Related Documents**

147

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

151

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

158

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

160

161 **4.4 Normative References**

162

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

168

169 RFC 2119
170 W3C XML v1.0 Second Edition Specification
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 DC 128 GUID

176

177 **4.5 Document Conventions**

178

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

182

183 The following conventions are used throughout this document:

184

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

188 **5 Design Objectives**

189 **5.1 Problem Description & Goals for ebXML**

190 For over 25 years *Electronic Data Interchange (EDI)* has given companies the prospect
191 of eliminating paper documents, reducing costs, and improving efficiency by exchanging
192 business information in electronic form. Ideally, companies of all sizes could conduct
193 *eBusiness* in a completely ad hoc fashion, without prior agreement of any kind. But this
194 vision has not been realized with *EDI*; only large companies are able to afford to
195 implement it, and much *EDI*-enabled *eBusiness* is centered around a dominant enterprise
196 that imposes proprietary integration approaches on its trading partners.

197
198
199 In the last few years, the *Extensible Markup Language (XML)* has rapidly become the
200 first choice for defining data interchange formats in new *eBusiness* applications on the
201 Internet. Many people have interpreted the XML groundswell as evidence that "*EDI* is
202 dead" – made completely obsolete by the XML upstart -- but this view is naïve from both
203 business and technical standpoints.

204
205 *EDI* implementations encode substantial experience in business processes, and companies
206 with large investments in *EDI* integration will not abandon them without good reason.
207 XML might enable more open, more loosely-coupled, and more object- or component-
208 oriented systems than *EDI*. XML might enable more flexible and innovative
209 "eMarketplace" business models than *EDI*. But the challenges of designing messages
210 that meet business process requirements and standardizing their semantics are
211 independent of the syntax in which the messages are encoded.

212
213 The ebXML specifications provide a framework in which *EDI*'s substantial investments
214 in business processes can be preserved in an architecture that exploits XML's new
215 technical capabilities.

216 **5.2 Caveats and Assumptions**

217 This specification is designed to provide a high level overview of ebXML, and as such,
218 does not provide the level of detail required to build ebXML applications, components,
219 and related services. Please refer to each of the respective ebXML Project Team
220 Specifications to get the level of detail.

221 **6 ebXML System Overview**

222
223 Figure 1 below shows a high level conceptual model for two *Trading Partners*, first
224 configuring and then engaging in a simple business transaction and interchange. This
225 model is provided as an example of the process and steps that MAY be REQUIRED to
226 configure and deploy ebXML applications and related system components. These
227 components MAY be implemented in an incremental manner. The ebXML specifications

231 are not limited to this simple model, provided here as quick introduction to the concepts.
232 Specific ebXML implementation examples are described in Appendix A.

233

234 The conceptual overview described below introduces the following concepts and
235 underlying architecture:

236

- 237 1. A standard mechanism for describing a *Business Process* and its associated
238 information model.
- 239 2. A mechanism for registering and storing a *Business Process and Information*
240 *Meta Model* so that it can be shared/reused.
- 241 3. Discovery of information about each participant including:
 - 242 • The *Business Processes* they support.
 - 243 • The *Business Service Interfaces* they offer in support of the *Business*
244 *Process*.
 - 245 • The *Business Messages* that are exchanged between their respective
246 *Business Service Interfaces*.
 - 247 • The technical configuration of the supported transport, security and
248 encoding protocols.
- 249 4. A mechanism for registering the aforementioned information so that it MAY be
250 discovered and retrieved.
- 251 5. A mechanism for describing a mutually agreed upon business arrangement which
252 MAY be derived from information provided by each participant from item 3
253 above.
- 254 6. A standardized business *Messaging Service* that enables interoperable, secure and
255 reliable exchange of messages between two parties.
- 256 7. A mechanism for configuration of the respective *Messaging Services* to engage in
257 the agreed upon *Business Process* in accordance with the constraints defined in
258 the business arrangement.

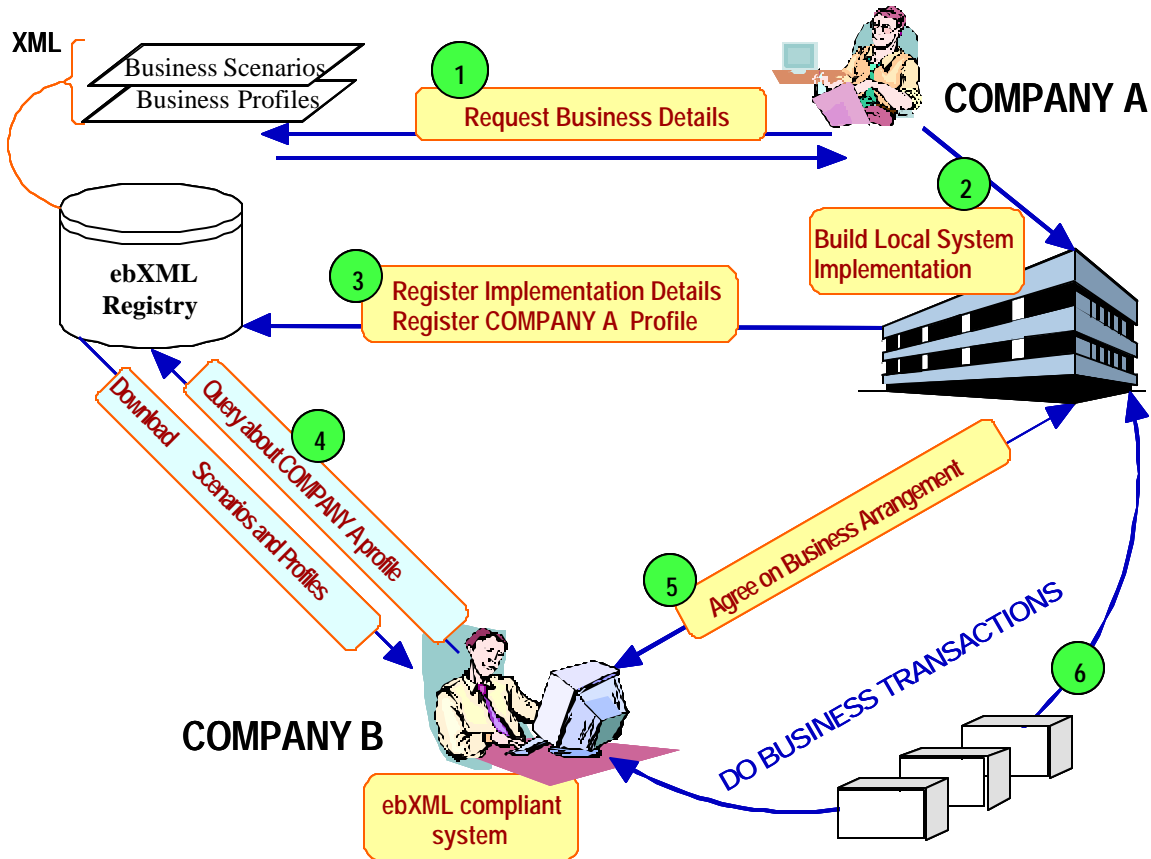


Figure 1 - a high level overview of the interaction of two companies conducting eBusiness using ebXML.

259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281

In Figure 1, Company A has become aware of an *ebXML Registry* that is accessible on the Internet (Figure 1, step 1). Company A, after reviewing the contents of the *ebXML Registry*, decides to build and deploy its own ebXML compliant application (Figure 1, step 2). It SHOULD be noted that custom software development is not a necessary prerequisite for ebXML participation. ebXML compliant applications and components MAY also be commercially available as shrink-wrapped solutions.

Company A then submits its own implementation details, reference links, and *Business Profile information* to the *ebXML Registry* (Figure 1, step 3). The business profile submitted to the *ebXML Registry* describes the company's ebXML capabilities and constraints, as well as its supported business processes. These business scenarios are XML versions of the *Business Processes* and associated information parcels (e.g. a sales tax calculation) that the company is able to engage in. After receiving verification that the format and usage of a business scenario is correct, an acknowledgment is sent to Company A by the *ebXML Registry* (Figure 1, step 3).

Company B discovers the business scenarios supported by Company A in the ebXML Registry (Figure 1, step 4). Company B sends a request to Company A stating that they would like to engage in a business transaction using ebXML (Figure 1, step 5). Company

282 B acquires a shrink-wrapped application that is ebXML compliant. Company A knows
 283 that its business scenarios and profiles are compliant with the ebXML infrastructure
 284 based on the information available in the ebXML specifications.

285

286 Before engaging in that the scenario Company B submits a proposed business
 287 arrangement directly to Company A's ebXML compliant software interface. The
 288 proposed business arrangement outlines the mutually agreed upon business scenarios and
 289 specific agreements on who it wants to conduct business transactions with Company A.
 290 The business arrangement also contains information pertaining to the messaging
 291 requirements for transactions to take place, contingency plans, and security-related
 292 requirements (Figure 1, step 5). Company A accepts the business agreement which then
 293 triggers an acknowledgement message that is sent directly to Company B's ebXML
 294 software application (Figure 1, step 5). Company A and B are now ready to engage in
 295 *eBusiness* using ebXML (Figure 1, step 6).

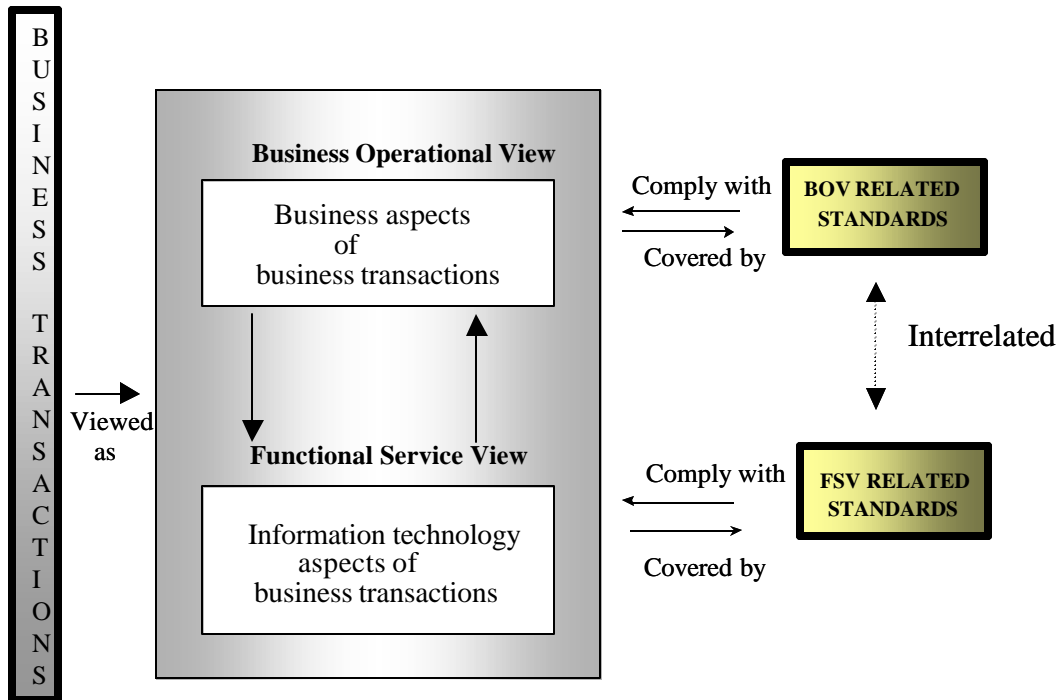
296 **7 ebXML Architecture Reference Model**

297

298 **7.1 Overview**

299 The *ebXML Architecture Reference Model* uses the following two views to describe the
 300 relevant aspects of *eBusiness* transactions. This model is based upon the Open-edi
 301 Reference Model, ISO 14662.

302



303

304

305

306 The ebXML architecture is broken down into the *Business Operational View (BOV)* and
 307 the supporting *Functional Service View (FSV)* described above. The assumption for

308 ebXML is that the *FSV* serves as a reference model that MAY be used by commercial
309 software vendors to help guide them during the development process. The underlying
310 goal of the *ebXML Reference Model* is to provide a clear distinction between the
311 operational and functional views, so as to ensure the maximum level of system
312 interoperability and backwards compatibility with legacy systems (when applicable). As
313 such, the resultant *BOV*-related standards provide the business and object class models
314 needed to construct ebXML compliant applications and components.

315

316 While business practices from one organization to another are highly variable, most
317 activities can be decomposed into *Business Processes* which are more generic to a
318 specific type of business. This analysis through the modeling process will identify object
319 classes and models that are likely candidates for standardization. The ebXML approach
320 looks for standard reusable components from which to construct interoperable ebXML
321 applications and components. The *BOV* and *FSV* are described in more detail below.

322

323 The *BOV* addresses:

324

325 a) The semantics of business data in transactions and associated data interchanges

326

327 b) The architecture for business transactions, including:

328

- 329 • operational conventions;
- 330 • agreements and arrangements;
- 331 • mutual obligations and requirements.

332

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

334

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

337

- 338 • Functional capabilities;
- 339 • *Service Interfaces*;
- 340 • Protocols and *Messaging Services*.

341

342 This includes, but is not limited to:

343

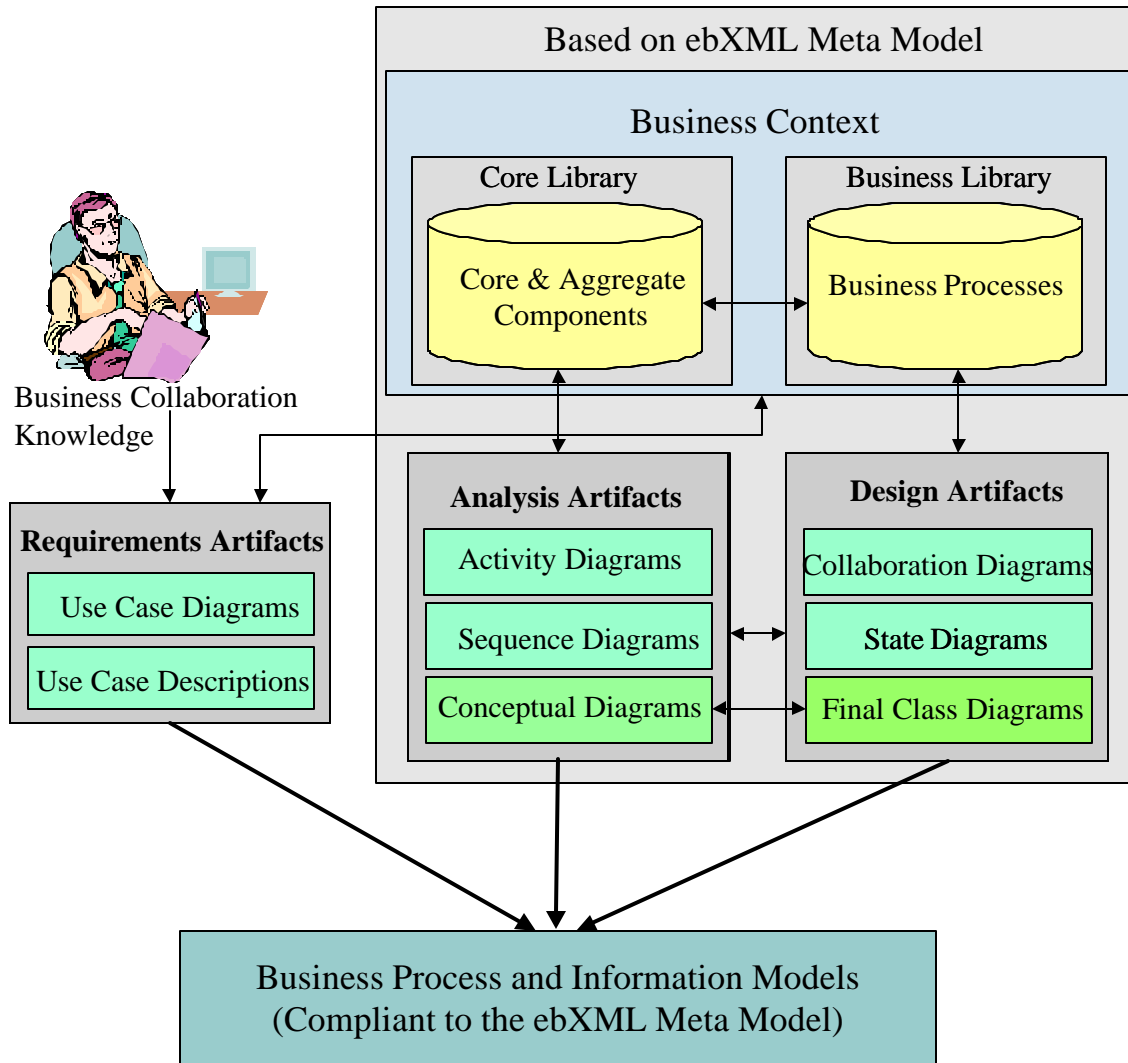
- 344 • Capabilities for implementation, discovery, deployment and run time scenarios;
- 345 • User *Application* interfaces;
- 346 • Data transfer infrastructure interfaces;
- 347 • *Protocols* for enabling interoperability of XML vocabulary deployments from
348 different organizations.

349

350 **7.2 ebXML Business Operational View**

351

352 The modeling techniques described in this section are not mandatory requirements for
 353 participation in ebXML compliant business transactions. Figure 3 below provides a
 354 detailed view of the ebXML BOV.
 355



356
 357
 358
 359

Figure 3 - the Business Operational View

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

366
 367

368 The first phase defines the requirements artifacts that describe the problem using *Use*
 369 *Case Diagrams and Descriptions*. If *Core Library* entries are available from an ebXML

370 compliant *Registry* they will be utilized, otherwise new *Core Library* entries will be
371 created and registered in an ebXML compliant *Registry*.

372

373 The second phase (analysis) will create activity and sequence diagrams describing the
374 *Business Processes*. *Class Diagrams* will capture the associated information parcels
375 (business messages). The analysis phase reflects the business knowledge contained in the
376 *Core Library*. No effort is made to force the application of object-oriented principles. The
377 class diagram is a free structured data diagram.

378

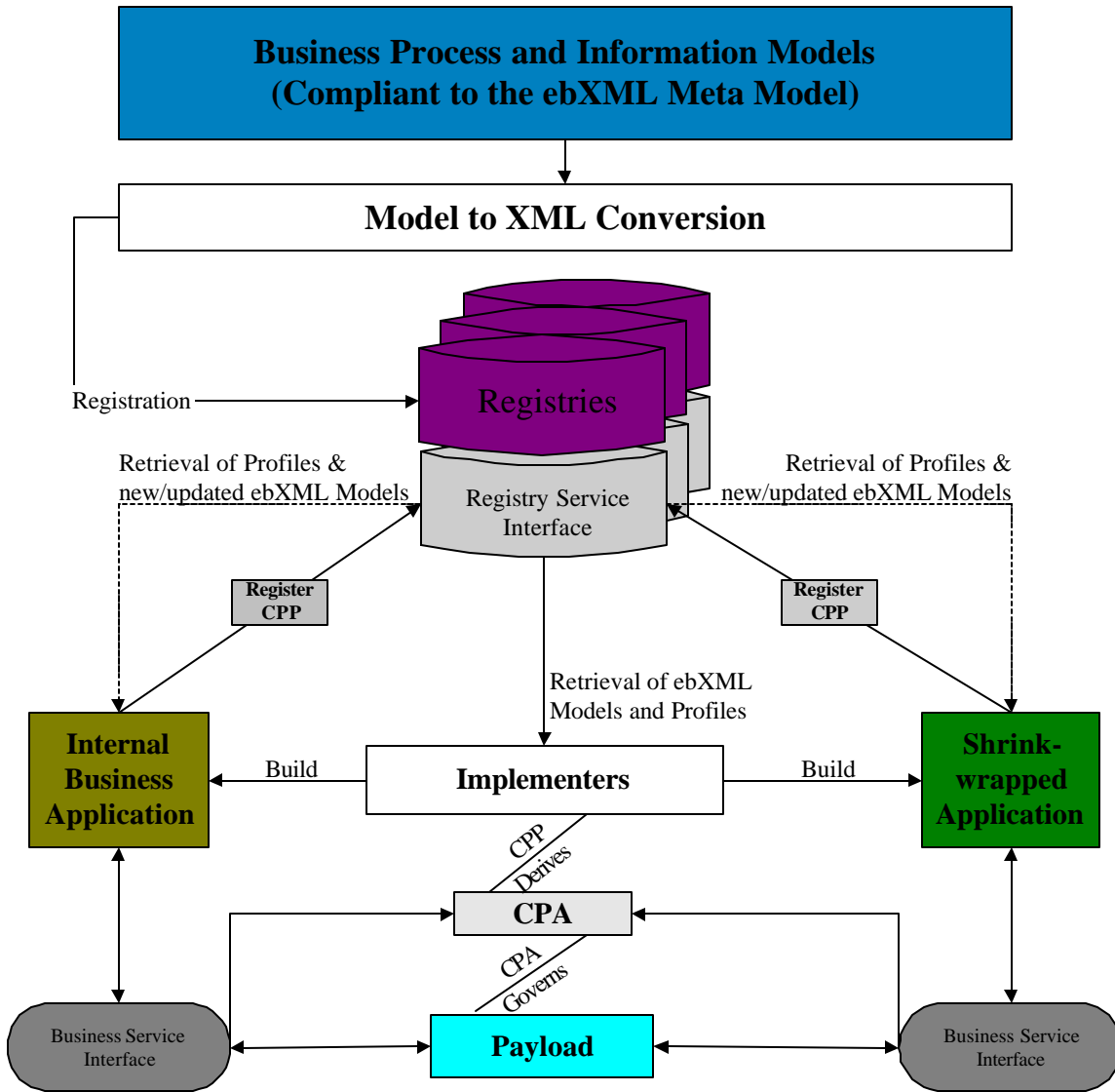
379 The design phase is the last step of standardization, which MAY be accomplished by
380 applying object-oriented principles. In addition to generating collaboration diagrams, a
381 state diagram MAY also be created. The class view diagram from the analysis phase will
382 undergo harmonization to align it with other models in the same industry and across
383 others.

384

385 In ebXML interoperability is achieved by applying *Business Objects* across all class
386 models. The content of the *Business Library* is created by analyzing existing *Business*
387 *Objects* as used by many industries today in conjunction with the *Core Library* content
388 and ebXML selected modeling methodology.

389

390 **7.3 ebXML Functional Service View**



391
392
393 **Figure 4 - ebXML Functional Service View**
394

395 As illustrated in Figure 4 above, the *ebXML Registry* system is an important part of
396 ebXML. It serves as the storage facility for the *Business Process and Information Meta*
397 *Models* developed by industry groups, *SMEs*, and other organizations. In order to store
398 the models in the *Registry*, they are converted to XML (e.g. XML DTD, Schema, etc.).
399 This XML-based business information SHALL be expressed in a manner that will allow
400 discovery down to the atomic data level via a consistent methodology. In order to enable
401 this functionality, the use of *Unique Identifiers (UIDs)* is REQUIRED for all items within
402 an *ebXML Registry System*. *UID* keys are REQUIRED references for all ebXML content.
403 *Globally Unique Identifiers (DC 128 - GUID)* MAY be used to ensure that *Registry*
404 entries are truly globally unique, and thus when systems query a *Registry* for a *GUID*,
405 one and only one result SHALL be retrieved.

406

407 To facilitate semantic recognition of *Business and Information Meta Models*, the *Registry*
408 system SHALL provide a mechanism for incorporating human readable descriptions for
409 *Registry* items. Existing *Business Process and Information Models* (e.g. RosettaNet PIPs)
410 SHALL be assigned *UID* keys when they are registered in an ebXML compliant *Registry*
411 system. These *UID* keys MAY be implemented in physical *XML* syntax in a variety of
412 ways. These mechanisms include, but are not limited to:

413

- 414 • A pure explicit reference mechanism (example: URN:*UID* method),
- 415 • A referential method (example: URI:*UID* / namespace:*UID*),
- 416 • An object-based reference compatible with W3C Schema (*example*
417 URN:complextype name), and
- 418 • A datatype based reference (example: ISO 8601:2000 Date/Time/Number
419 datatyping and then legacy datatyping).

420

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

426

427 The underlying ebXML Architecture is distributed in such a manner to minimize the
428 potential for a single point of failure within the ebXML infrastructure. This specifically
429 refers to *Registry* and *Repository Services* (see *Registry* and *Repository Functionality*,
430 Section 9.4 for details of this architecture).

431 **8 ebXML Functional Phases**

432

433 **8.1 Overview**

434

435 **8.1.1 The Implementation Phase**

436 The implementation phase deals specifically with the procedures for creating an
437 application of the ebXML infrastructure.

438

439 **8.1.2 The Discovery and Retrieval Phase**

440 The Discovery and Retrieval Phase covers all aspects of actual discovery of ebXML
441 related resources and self enabled into the ebXML infrastructure.

442

443 **8.1.3 The Run Time Phase**

444 The Run Time phase covers the execution of an ebXML scenario with the actual
445 associated ebXML transactions.

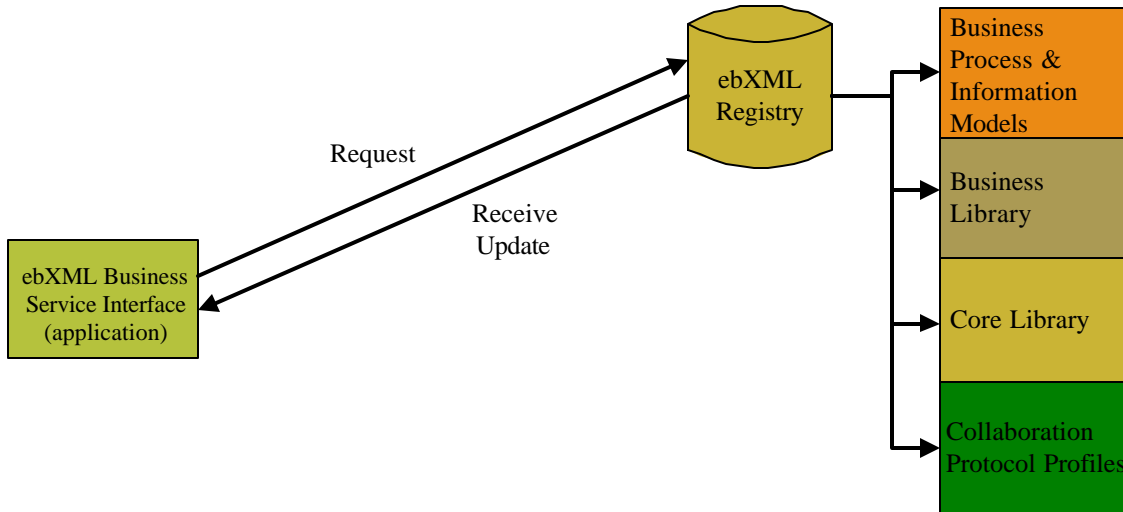
446

447 **8.2 Implementation Phase**

448

449 A *Trading Partner* wishing to engage in an ebXML compliant transaction, must first
450 request a copy of the ebXML specification. The Specification is then downloaded to the

451 *Trading Partner*(via HTTP, FTP, etc.). The *Trading Partner* studies the ebXML
 452 specification and subsequently requests to download the *Core Library* and the *Business*
 453 *Object Library*. The *Trading Partner* MAY also request other *Trading Partners*'
 454 *Business Process* information (stored in it business profile) for analysis and review. The
 455 *Trading Partner* MAY also submit its own *Business Process* information to an ebXML
 456 compliant *Registry* system.
 457
 458 Figure 5 below, illustrates a basic interaction between an ebXML *Registry* system and a
 459 *Business Service Interface*.

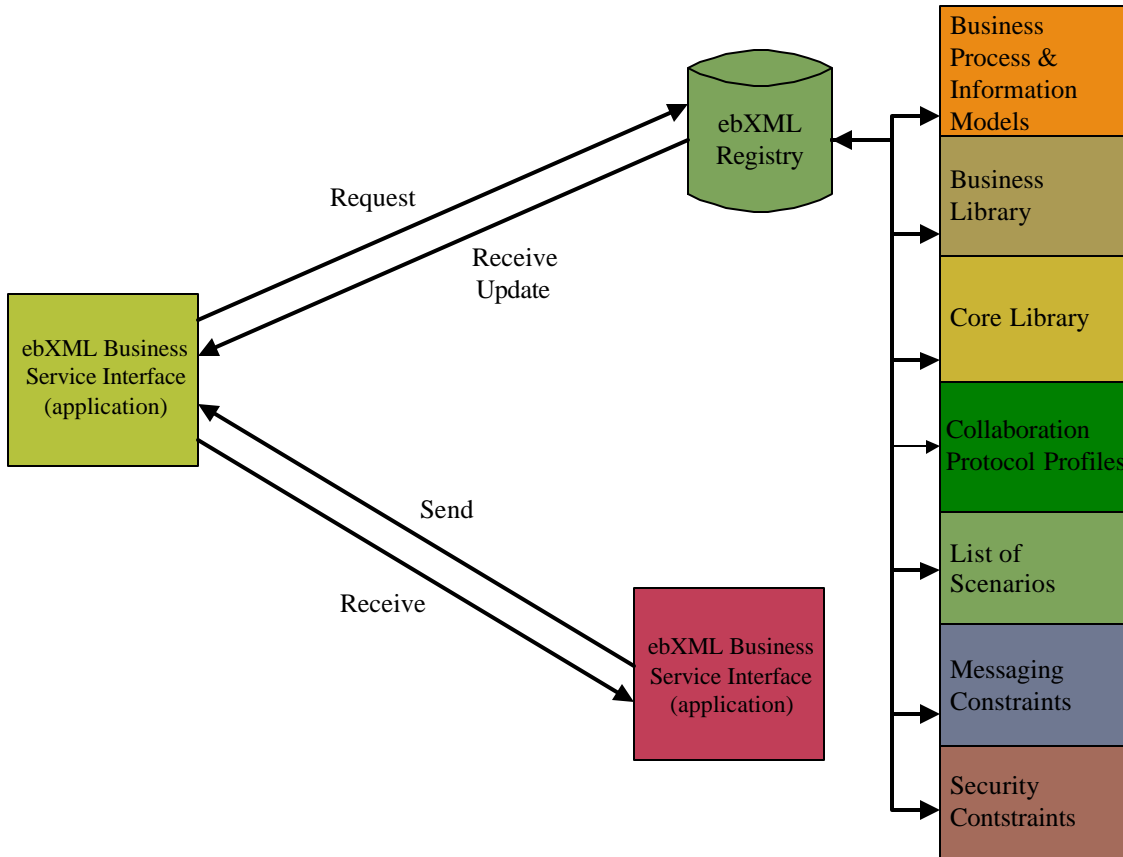


460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473

Figure 5 - Functional Service View: Implementation Phase

8.3 Discovery and Retrieval Phase

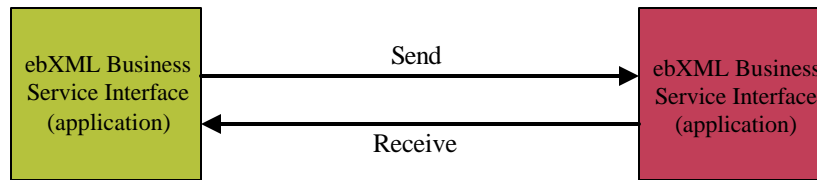
A *Trading Partner* who has implemented an ebXML *Business Service Interface* can now begin the process of discovery and retrieval (Figure 6 below). One possible discovery method MAY be to request the *Trading Partner Profile* of another *Trading Partner*. Requests for updates to *Core Libraries*, *Business Object Libraries* and updated or new *Business Process* and information models are also methods that SHALL be supported by an ebXML *Application*. This is the phase where *Trading Partners* discover the semantic meaning of business information being requested by other *Trading Partners*.



474
475
476 **Figure 6 - Functional Service View: Discovery and Retrieval Phase**
477

478 **8.4 Run Time Phase**
479

480 In the Run Time Phase, ebXML messages are being exchanged between *Trading*
481 *Partners* utilizing the *ebXML Messaging Service*. If it becomes necessary to make calls to
482 the *Registry* during the Run Time, this will be considered as a reversion to the Discovery
483 and Retrieval Phase.
484



485
486
487 **Figure 7 - Functional Service View: Run Time Phase**
488

489 **9 ebXML Infrastructure**

490

491 **9.1 Trading Partner Information [CPP and CPA's]**

492

493 **9.1.1 Introduction**

494 To facilitate the process of conducting *eBusiness*, *SMEs* and other organizations need a
495 mechanism to publish information about the *Business Processes* they support along with
496 specific technology implementation details about their capabilities for exchanging
497 business information. This is accomplished through the use of a *Collaboration Protocol*
498 *Profile (CPP)*. The CPP is a document which allows a *Trading Partner* to express their
499 minimum *Business Process* and *Business Service Interface* requirements in a manner
500 where they can be universally understood by other ebXML compliant *Trading Partners*.

501

502 To facilitate the process of conducting electronic business, organizations also need a
503 mechanism to publish information about the *Business Processes* they support, along with
504 specific technology details about their capabilities for sending and receiving business
505 documents. ebXML defines the ability for this to be realized under the broad notion of a
506 *Collaboration Protocol Agreement (CPA)*. A CPA is a document that represents the
507 intersection of two CPP's and is mutually agreed upon by both Trading Partners who
508 wish to conduct *eBusiness* using ebXML.

509

510 **9.1.2 CPP Formal Functionality**

511 The *CPP* describes the specific capabilities that a *Trading Partner* supports as well as the
512 *Service Interface* requirements that need to be met in order to exchange business
513 documents with that *Trading Partner*. Each *Trading Partner* SHALL register one and
514 only one *CPP* in an ebXML compliant Registry system. The CPP contains essential
515 information about the Trading Partner, which MAY include (but is not limited to):
516 contact information, industry classification, supported business processes, and interface
517 requirements.

518

519 *CPP's* describe the *Business Processes* that a given *Trading Partner* supports, plus the
520 *Messaging Service* interface requirements that the given *Trading Partner* will use as a
521 support mechanism for such collaborations. *CPP's* MAY optionally include security and
522 other implementation specific details. Each ebXML compliant *Trading Partner* SHALL
523 register their *CPP* in an ebXML compliant *Registry* system, thus providing a discovery
524 mechanism that allows *Trading Partners* to (1) find one another, (2) discover the
525 *Business Process* that other *Trading Partners* support.

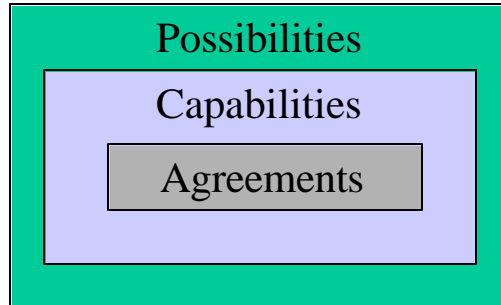
526

527 **9.1.3 CPA Formal Functionality**

528 A *Collaboration Protocol Agreement (CPA)* describes: (1) the *Messaging Service*
529 (technology), and (2) the *Business Process* (application) requirements that are agreed
530 upon by two or more *Trading Partners*. Conceptually, ebXML supports a three level
531 view of narrowing subsets to arrive at *CPA's* for transacting *eBusiness*. The outer-most
532 scope relates to all of the possibilities that a *Trading Partner* could do, with a subset of
533 that of what a *Trading Partner* is capable of doing, with a subset of what a *Trading*
534 *Partner* "will" do.

535
536
537
538
539
540

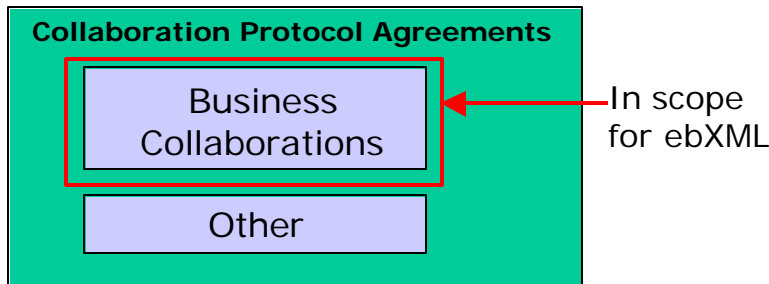
A *CPA* contains the *Messaging Service* interface requirements as well as the implementation details pertaining to the mutually agreed upon *Business Processes* that both *Trading Partners* agree to use to conduct *eBusiness*. *Trading Partners* MAY decide to register their *CPA*'s in an ebXML compliant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



541
542
543
544
545
546
547
548
549
550

Figure 8 - Three level view of CPA's

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



551
552
553
554
555

Figure 9 - Scope for CPA's

9.1.4 CPP Interfaces

556
557
558
559
560
561
562
563
564
565

Business Process: The CPP must be capable of referencing one or more business processes supported by the entity owning the CPP instance. The CPP must also reference the Roles within that BP that the user is capable of assuming.

The CPP must be capable of referencing, either directly or indirectly, the CPA for each supported Business Process.

The CPP must be capable of being stored and retrieved from a Registry Mechanism

566

567 The CPP must be capable of being carried in the payload of the ebXML Messaging
568 service. A CPP may also describe binding details that are used to build an ebXML
569 message header.

570

571 **9.1.5 CPA Interfaces**

572

573 A CPA has an Interface to a software component used by a Trading Partner via the
574 ebXML Messaging mechanism.

575

576 A CPA has an interface to the CPP by the fact it must narrow down the Trading Partners
577 Capabilities into what the Trading Partner “will” do. What a Trading Partner “will” do
578 must be within that Trading Partners’ capabilities hence an abstract interface between the
579 two documents.

580

581 A CPA has an interface to a Business Process document by the fact it may be reached and
582 referenced for each Business Process.

583

584 A CPA also may be stored in a Registry mechanism, hence an implied ability to be stored
585 and retrieved is present.

586

587 **9.1.6 Non-Normative Implementation Details [CPP and CPA's]**

588

589 A *CPA* is negotiated after the discovery process and is essentially a snapshot of the
590 *Messaging Services* and *Business Process* related information that two or more *Trading*
591 *Partners* agree to use to exchange business information. If any parameters contained
592 within an accepted *CPA* change after the agreement has been executed, a new *CPA*
593 SHALL be negotiated between all parties.

594

595 **9.2 Business Process and Information Modeling**

596

597 **9.2.1 Introduction**

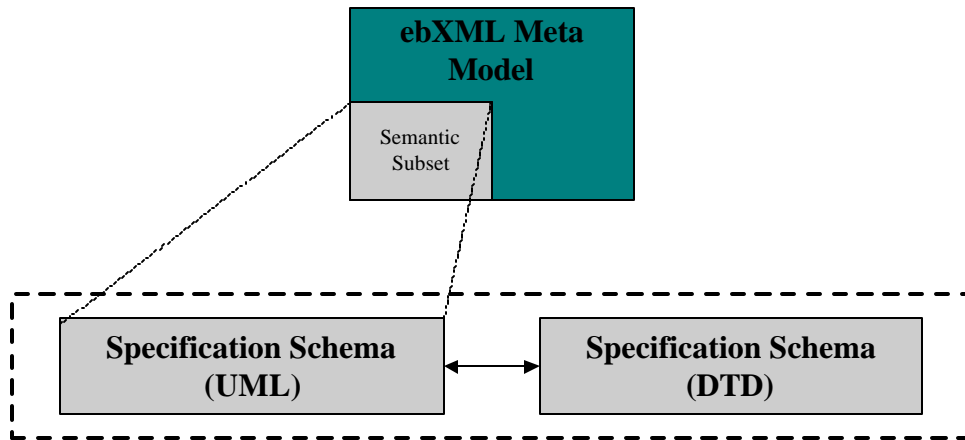
598 The ebXML *Business Process and Information Meta Model* is a mechanism that allows
599 *Trading Partners* to capture the details for a specific business scenario using a consistent
600 modeling methodology. A *Business Process* describes in detail how *Trading Partners*
601 take on roles, relationships and responsibilities to facilitate interaction with other *Trading*
602 *Partners* in shared *Business Processes*. The interaction between roles takes place as a
603 choreographed set of *Business Transactions*. Each *Business Transaction* is expressed as
604 an exchange of electronic *Business Documents*. The sequence of the exchange is defined
605 by the *Business Process*, messaging and security considerations. *Business Documents* are
606 composed from re-useable business information components (see “Relationships to Core
607 Components” under 9.2.3 “Interfaces” below). At a lower level, *Business Processes* can
608 be composed of re-useable *Core Processes*, and *Business Objects* can be composed of re-
609 useable *Core Components*.

610

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

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

623
 624 The relationship between the ebXML *Business Process and Information Meta Model* and
 625 the ebXML *Specification Schema* can be shown as follows:
 626



627
 628
 629
 630

Figure 10 - ebXML Meta Model - Semantic Subset

631 The *Specification Schema* supports the specification of *Business Transactions* and the
 632 choreography of *Business Transactions* into *Business Collaborations*. Each *Business*
 633 *Transaction* can be implemented using one of many available standard patterns. These
 634 patterns determine the actual exchange of messages and signals between the partners to
 635 achieve the required legally binding electronic commerce transaction. To help specify the
 636 patterns the *Specification Schema* is accompanied by a set of standard patterns, and a set
 637 of modeling elements common to those standard patterns. The full specification, thus, of
 638 a business process consists of a business process model specified against the
 639 *Specification Schema* and an identification of the desired pattern(s). This full
 640 specification is then the input to the formation of *Trading Partner Collaboration Profiles*
 641 and *Collaboration Agreements*. This can be shown as follows:
 642

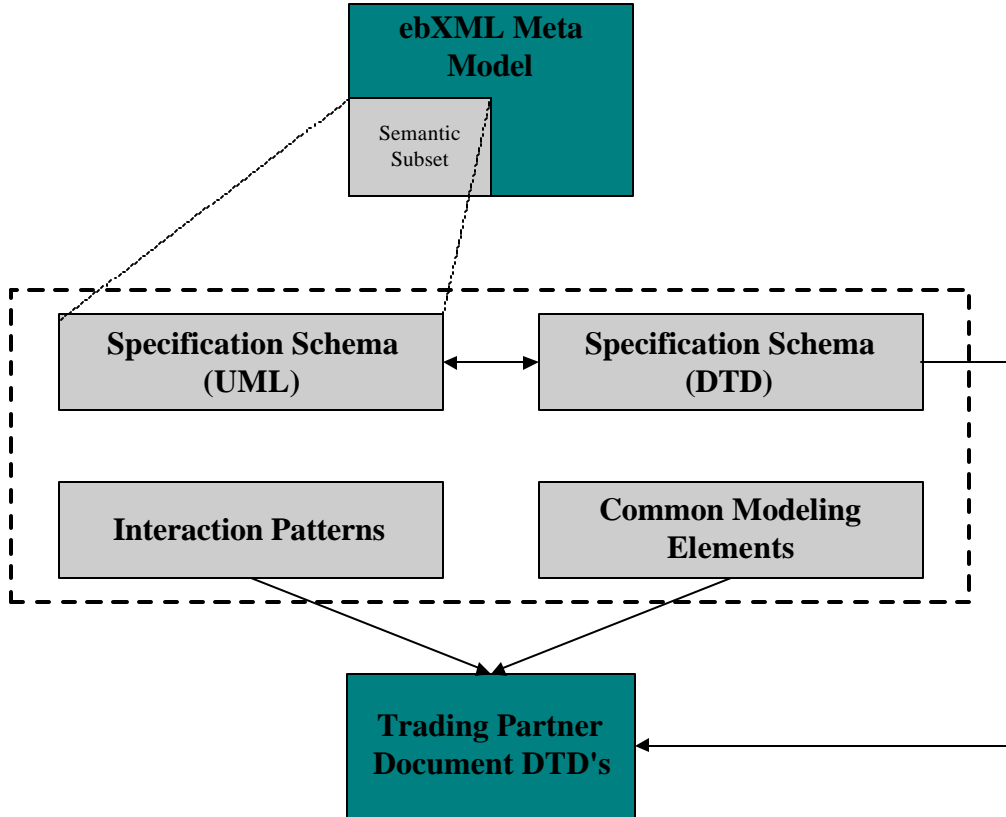


Figure 11 - ebXML Meta Model

643
 644
 645
 646
 647
 648
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 660
 661
 662
 663
 664
 665
 666
 667

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*.

To further facilitate the creation of consistent business processes and information models, ebXML will define a core set of *Business Processes* in parallel with a *Core Library*. It is possible that users of the ebXML infrastructure MAY wish to extend this set or use their own *Business Processes*.

9.2.2 Formal Functionality

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

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

To be understood by an application, a *Business Process* SHALL be expressible in XML syntax. A *Business Process* SHALL be comprised of an information model or XML-

668 based representation of that model, that is capable of expressing the following types of
669 information:

- 670 • Choreography for the exchange of document instances.
- 671 • References to *Metadata* (possibly *DTD's* or *Schemas*) that add structure to
672 business data.
- 673 • Definition of the roles for each participant in a *Business Process*.
- 674 • May reference supporting *Metadata*.
- 675 • Provide a context constraint that affects *Core Components*
- 676 • Provide the framework for establishing *CPAs*
- 677 • The domain owner of the *Business Process* and contact information.

678

679 [NOTE: this is not an inclusive list.]

680

681 **9.2.3 Interfaces**

682 The interface from a *Business Process* to its associated *Business Process and Information*
683 *Meta Model* to other parts of the ebXML Architecture, or to other tools and environments
684 is outside the scope of the ebXML specifications.

685

686 **Relationship to CPP and CPA**

687 The *CPP* instance of a *Trading Partner* defines that partner's functional and technical
688 capability to support zero, one, or more *Business Processes* and one or more roles in each
689 process.

690

691 The agreement between two *Trading Partners* defines the actual conditions under which
692 the two partners will conduct business transactions together. Accordingly, the interface
693 between the *Business Process* and its associated *Business Process and Information Meta*
694 *Model* and the *CPA* is the part of the *Business Process* document that is instantiated as an
695 XML document that represents the business transactional layer of the *Business Process*
696 *and Information Meta Model*. The expression of the sequence of commercial transactions
697 in XML is shared between the *Business Process* and *Trading Partner Information*
698 models.

699

700 **Relationship to Core Components**

701 A *Business Process* instance MAY specify the constraints for exchanging business data
702 with other *Trading Partners*. The business data MAY be comprised of components of
703 the ebXML *Core Library*. Accordingly, a *Business Process* document SHALL reference
704 the *Core Components* directly or indirectly using a XML document with metadata
705 (possibly *DTD's* or *Schemas*) that can be uniquely referenced. The mechanism for
706 interfacing with the *Core Components* and *Core Library* SHALL be by way of a *UID* or
707 *GUID* for each component.

708

709 **Relationship to ebXML Messaging**

710 A *Business Process* instance SHALL be capable of being transported from a *Registry*
711 mechanism to another *Registry* mechanism via an *ebXML Message*. It SHALL also be

712 capable of being transported between a *Registry* and a users application via the *ebXML*
 713 *Messaging Service*.

714

715 **Relationship to a Registry System**

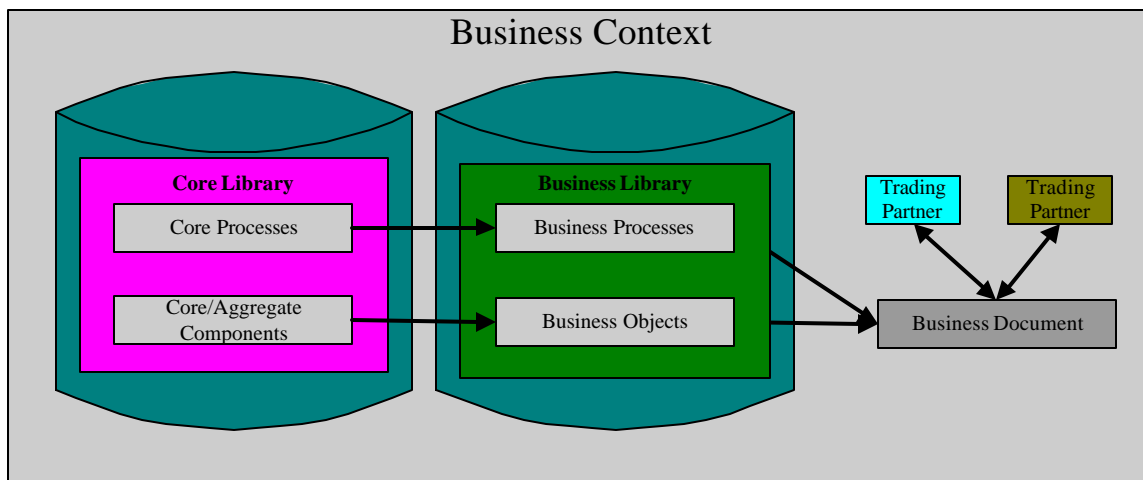
716 A *Business Process* instance intended for use within the ebXML infrastructure SHALL
 717 be retrievable through a Registry query, and therefore, each *Business Process* SHALL
 718 contain a *UID* or *GUID*.

719

720 **9.2.4 Non-Normative Implementation Details**

721 The exact composition of a *Business Object* or a *Business Document* is guided by a set of
 722 contexts derived from the *Business Process*. The modeling layer of the architecture is
 723 highlighted in green in Figure 12 below.

724



725

726

727 *Figure 12 – ebXML Business Process and Information Modeling layer*

728

729 ebXML *Business Process and Information Models* MAY be created following the
 730 RECOMMENDED ebXML *Modeling Methodology (UML)*, or MAY be arrived at in any
 731 other way, as long as they comply with the ebXML *Business Process and Information*
 732 *Meta Models*.

733

734 **9.3 Core Components and Core Library Functionality**

735

736 **9.3.1 Introduction**

737

738 A *Core Component* captures information about a real world business concept, and the
 739 relationships between that concept, other *Business Information*, and a contextual
 740 description that describes how a *Core* or *Aggregate Component* may be used in a
 741 particular ebXML *eBusiness* scenario.

742

743 A *Core Component* can be either an individual piece of business information, or a natural
 744 “go-together” family of *Business Information* that may be assembled into *Aggregate*
 745 *Components*.

746

747 The *ebXML Core Components Project Team* SHALL define an initial set of *Core*
748 *Components*. ebXML users may adopt and/or extend components from the ebXML *Core*
749 *Library*.

750

751 **9.3.2 Formal Functionality**

752

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

755

756 *Core Components* SHALL be storable and retrievable using an ebXML *Registry*
757 *Mechanism*.

758

759 *Core Components* SHALL capture and hold a minimal set of information to satisfy both
760 business and technical needs.

761

762 *Core Components* SHALL be expressible in XML syntax.

763

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

765

- 766 • Another *Core Component* in combination with one or more individual pieces of
767 *Business Information*.

768

- 769 • Other *Core Components* in combination with zero or more individual pieces of
770 *Business Information*.

771

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

773

774 **9.3.3 Interfaces**

775

776 A *Core Component* MAY be referenced indirectly or directly from a *Business Process*
777 instance. The *Business Process* MAY specify a single or group of core components as
778 required or optional information as part of a *Business Process*.

779

780 A *Core Component* MAY interface with a *Registry* mechanism by way of being storable
781 and retrievable in such a mechanism.

782

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

785

786

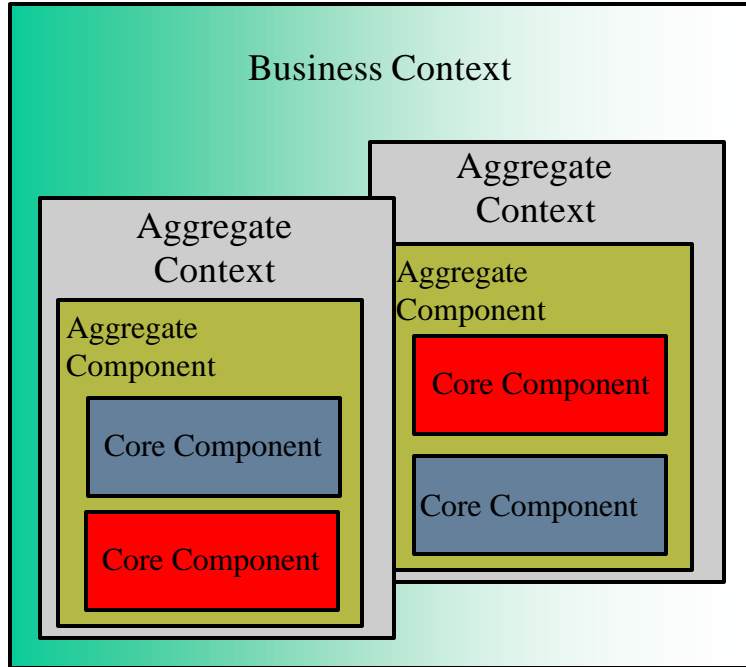
787 **9.3.4 Non-Normative Implementation Details**

788

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

791

792 The process of aggregating *Core Components* for a specific business context, shall
 793 include a means to identify the placement of a *Core Component* within another *Core*
 794 *Component*. It MAY also be a combination of structural contexts to facilitate *Core*
 795 *Component* re-use at different layers within another *Core Component* or *Aggregate*
 796 *Component*. This is referred to as *Business Context*.
 797
 798 Context MAY also be defined using the *Business Process and Information Meta Model*,
 799 which defines the instances in which the *Business Object* occurs.
 800



801
 802
 803 **Figure 13 - Business Context defined in terms of Aggregate Context and Aggregate and Core Components**
 804

805 The pieces of *Business Information*, or *Core Components*, within a generic *Core*
 806 *Component* may be either mandatory, or optional. A *Core Component* in a specific
 807 context or combination of contexts (aggregate or business context) may alter the
 808 fundamental mandatory/optional cardinality.
 809

810 **9.4 Registry Functionality**

811
 812 **9.4.1 Introduction**

813 An *ebXML Registry* provides a set of services that enable the sharing of information
 814 between users. A *Registry* is a component that maintains an interface to metadata for a
 815 registered item. Access to an *ebXML Registry* is provided through interfaces (APIs)
 816 exposed by *Registry Services*.
 817
 818

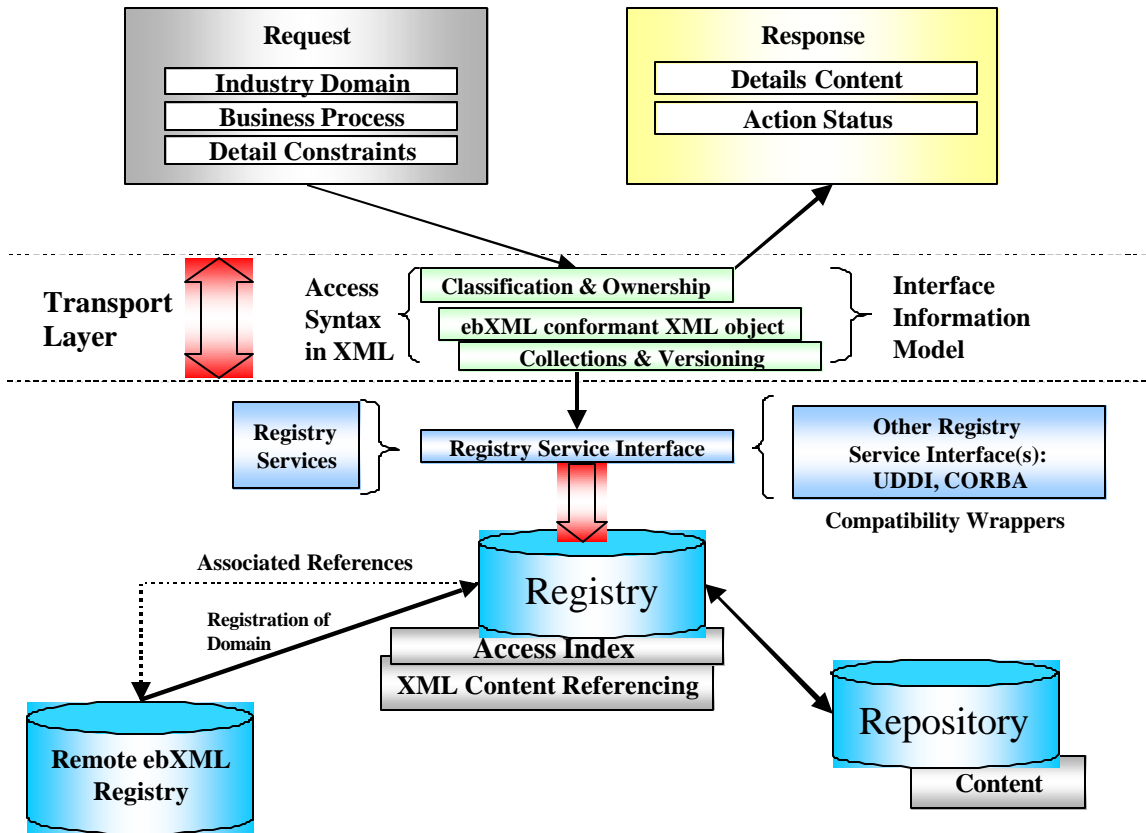


Figure 14 - Overall Registry / Repository Architecture.

819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840

9.4.2 Formal Functionality

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

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

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

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

841 The *Registry Interface* provides an application-to-registry automated access. Human-to-
842 Registry interactions SHALL be built as a layer over a *Registry Client* (e.g. a Web
843 browser) and not as a separate interface.

844

845 The *Registry* interface SHALL be designed to be transport layer neutral.

846

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

848

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

859

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

863

864 *Registry* messages SHALL exist to create, modify, and delete *Registry Items* and their
865 metadata.

866

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

869

870 9.4.3 Interfaces

871

872 **ebXML Messaging :**

873 The query syntax used by the *Registry* access mechanisms is independent of the physical
874 implementation of the backend system. The ebXML *Messaging Service* serves as the
875 transport mechanism for all communications into and out of the *Registry*.

876

877 **Business Process:**

878 Business Processes MAY be published and retrieved via ebXML *Registry* services.

879

880 **Core Components:**

881 *Core Components* MAY be published and retrieved via ebXML *Registry* services.

882

883 **Any item with metadata:** XML elements provide standard metadata about the item
884 being managed through ebXML *Registry* services. [NOTE: The metadata is separate from
885 the item itself, thus allowing the *ebXML Registry* to catalog arbitrary items.] Since

886 ebXML Registries are distributed each *Registry* MAY interact with and cross-reference
887 another ebXML *Registry*.

888

889 **9.4.4 Non-Normative Implementation Details**

890 The *Business Process and Information Model* within a *Registry* MAY be stored
891 according to various classification schemes.

892

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

895

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

898

899 Examples of extended Registry services functionality MAY be deferred to a subsequent
900 phase of the ebXML initiative. This MAY include: transformation services, workflow
901 services, quality assurance services and extended security mechanisms.

902

903 A *Registry* service MAY have multiple deployment models as long as the *Registry*
904 interfaces are ebXML compliant.

905

906 The assignment of a *GUID* to *Registry Items* MAY benefit from the use of a standard
907 algorithm such as the DC 128 GUID algorithm.

908

909 The *Business Process and Information Model* for an *ebXML Registry* service MAY be an
910 extension of the existing *OASIS Registry Information Model*, specifically tailored for the
911 storage and retrieval of business information, whereas the OASIS model is a superset
912 designed for handling extended and generic information content.

913

914 **9.5 Messaging Service Functionality**

915

916 **9.5.1 Introduction**

917 The *ebXML Message Service* mechanism SHALL provide a standard way to exchange
918 business messages among ebXML *Trading Partners*. The *ebXML Messaging Service*
919 provides a reliable means to exchange business messages without relying on proprietary
920 technologies and solutions. An *ebXML Message* contains structures for a *Header*
921 (necessary for routing and delivery) and a *Payload* section (necessary for transport).

922

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

927

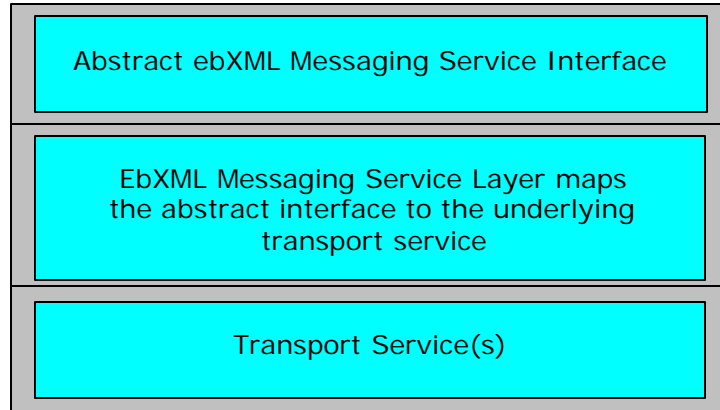


Figure 15 - ebXML Messaging Service

928
 929
 930
 931
 932
 933
 934
 935
 936
 937

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.

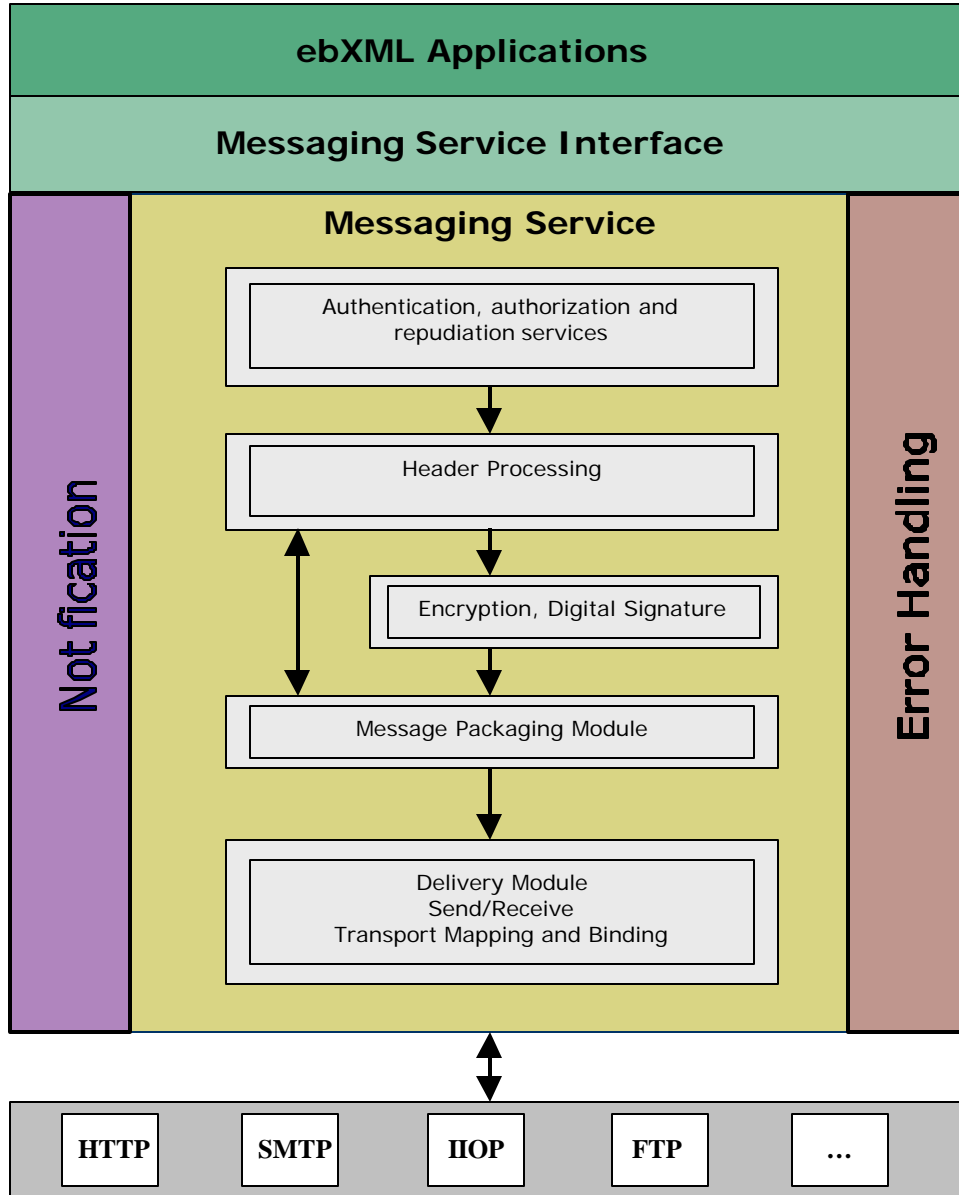


Figure 16 - The Messaging Service Architecture

938
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950

9.5.2 Formal Functionality

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

The *ebXML Messaging Service* SHALL prescribe formats for all messages between distributed ebXML *Components* including *Registry* mechanisms and compliant user *Applications*. It SHALL also utilize and enforce the "rules of engagement" defined in a *Collaboration Protocol Agreement (CPA)*.

951 The *ebXML Messaging Service* SHALL NOT place any restrictions on the content of the
952 payload.

953

954 The *ebXML Messaging Service* SHALL support simplex (one-way) and request/response
955 (either synchronous or asynchronous) message exchanges.

956

957 The *ebXML Messaging Service* SHALL meet business needs, consequently it SHALL
958 support sequencing of payloads in instances where multiple payloads or multiple
959 messages are being used.

960

961 The *ebXML Messaging Service Layer* SHALL enforce the "rules of engagement" as
962 defined by two parties in a *Collaboration Protocol Agreement* (including security and
963 *Business Process* functions related to message delivery). The *Collaboration Protocol*
964 *Agreement* defines the acceptable behavior by which each *Party* agrees to abide. The
965 definition of these ground rules can take many forms including formal *Collaboration*
966 *Protocol Agreements*, interactive agreements established at the time a business
967 transaction occurs (e.g. buying a book online), or other forms of agreement. There are
968 *Messaging Service Layer* functions that enforce these ground rules. Any violation of the
969 ground rules result in an error condition, which is reported using the appropriate means.

970

971 The *ebXML Messaging Service* SHALL perform all security related functions including:

972

- Identification
- Authentication (verification of identity)
- Authorization (access controls)
- Privacy (encryption)
- Integrity (message signing)
- Non-repudiation
- Logging

973

974

975

976

977

978

979

980

9.5.3 Interfaces

981

The *ebXML Message Service* provides ebXML with an abstract interface whose
982 functions, at an abstract level, include:

983

984

- Send – send an *ebXML Message* – values for the parameters are derived from the
985 *ebXML Message Headers*.
- Receive – indicates willingness to receive an *ebXML Message*.
- Notify – provides notification of expected and unexpected events.
- Inquire – provides a method of querying the status of the particular ebXML
988 Message interchange.

989

990

991

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

992

- Routing of received messages to internal systems

993

- Error notification

994

995 The *ebXML Messaging Service* SHALL help facilitate the interface to an ebXML
 996 *Registry*.

997

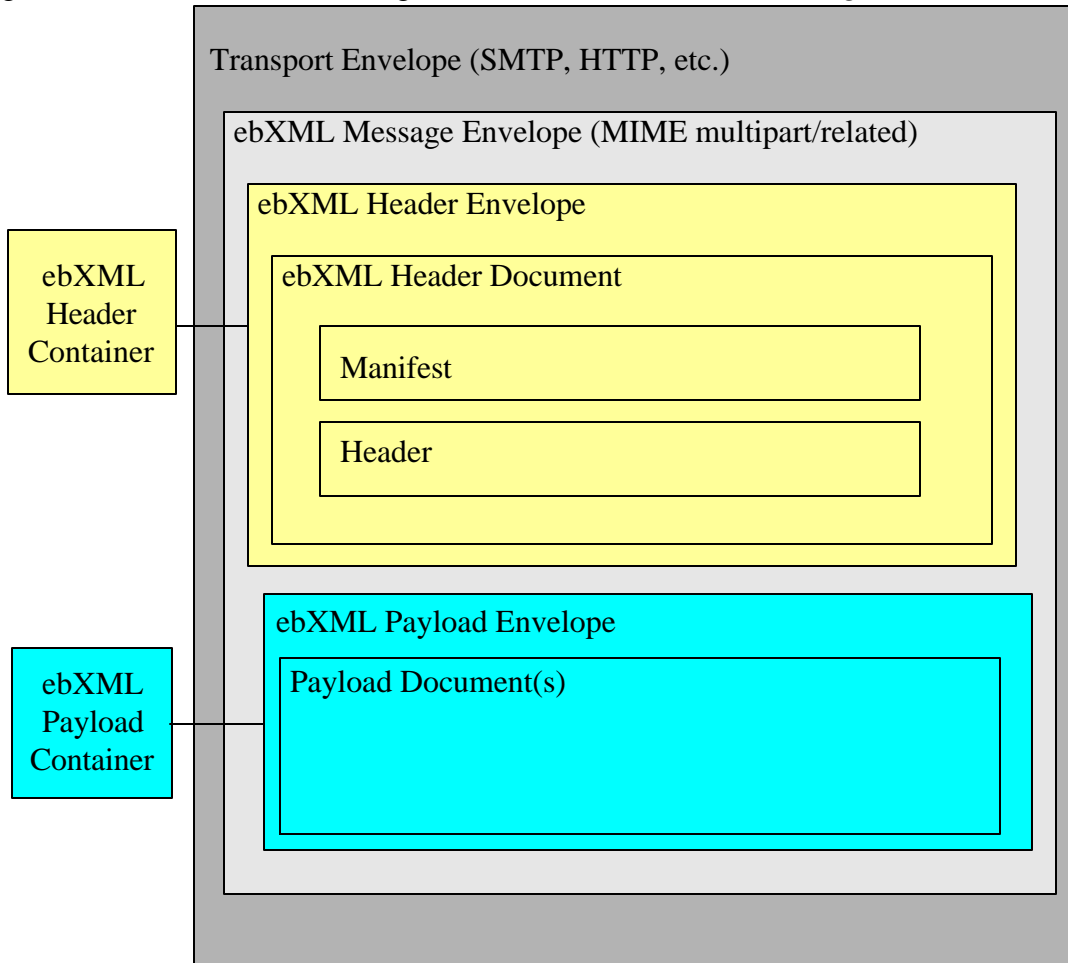
998 **9.5.4 Non-Normative Implementation Details**

999

1000 **ebXML Message Structure and Packaging**

1001

1002 Figure 17 below illustrates the logical structure of an *ebXML Message*.



1003

1004

1005

1006

Figure 17 - ebXML Message Structure

1007 An *ebXML Message* MAY consist of an OPTIONAL transport *Protocol* specific outer
 1008 *Communication Protocol Envelope* and a *Protocol* independent *ebXML Message*
 1009 *Envelope*. The *ebXML Message Envelope* MAY be packaged using the MIME
 1010 multipart/related content type. MIME is used as a packaging solution because of the
 1011 diverse nature of information exchanged between *Partners* in *eBusiness* environments.
 1012 For example, a complex B2B business transaction between two or more *Trading Partners*
 1013 might require a payload that contains an array of business documents (*XML* or other
 1014 document formats), binary images, or other related *Business Objects*.

1015 **10 Conformance**

1016 **10.1 Introduction**

1018
1019 This clause specified the general framework, concepts and criteria for *Conformance* to
1020 ebXML, including an overview of the conformance strategy for ebXML, guidance for
1021 addressing conformance in each ebXML technical specification, and the conformance
1022 clause specific to the Technical Architecture specification. Except for the Technical
1023 Architecture Specification, this clause does not define the conformance requirements for
1024 each of the ebXML technical specifications – the latter is the purview of the technical
1025 specifications.

1026
1027 The objectives of this section are to:

- 1028 a) Ensure a common understanding of conformance and what is required to claim
1029 conformance to this family of specifications;
- 1030 b) Ensure that conformance is consistently addressed in each of the component
1031 specifications;
- 1032 c) Promote interoperability and open interchange of *Business Processes* and
1033 messages;
- 1034 d) Encourage the use of applicable conformance test suites as well as promote
1035 uniformity in the development of conformance test suites.

1036
1037 Conformance to ebXML is defined in terms of conformance to the ebXML infrastructure
1038 and conformance to each of the technical specifications for ebXML. The primary
1039 purpose of conformance to ebXML is to increase the probability of successful
1040 interoperability between implementations and the open interchange of XML business
1041 documents and messages. While conformance is a necessary condition, it is not on its
1042 own a sufficient condition to guarantee interoperability. Successful interoperability and
1043 open interchange is more likely to be achieved if implementations conform to the
1044 requirements in the ebXML specifications.

1045 **10.2 Conformance to ebXML**

1046
1047 ebXML Conformance is defined as conformance to an ebXML system that is comprised
1048 of all the architectural components of the ebXML infrastructure and satisfies at least the
1049 minimum conformance requirements for each of the ebXML technical specifications,
1050 including the functional and interface requirements in this Technical Architecture
1051 specification.

1052
1053 In the context of ebXML, an implementation is said to exhibit conformance if it complies
1054 with the requirements of each applicable ebXML technical specification. The
1055 conformance requirements are stated in the conformance clause of each technical
1056 specification of ebXML. The conformance clause specifies explicitly all the
1057 requirements that have to be satisfied to claim conformance to that specification. These
1058 requirements MAY be applied and grouped at varying levels within each specification.
1059

1060

1061 **10.3 Conformance to the Technical Architecture Specification**

1062

1063 This section details the conformance requirements for claiming conformance to the
1064 Technical Architecture specification.

1065

1066 In order to conform to this specification, each ebXML technical specification:

1067

a) SHALL support all the functional and interface requirements defined in this
1068 specification that are applicable to that technical specification;

1069

b) SHALL NOT specify any requirements that would contradict or cause non-
1070 conformance to ebXML or any of its components;

1071

c) MAY contain a conformance clause that adds requirements that are more specific
1072 and limited in scope than the requirements in this specification;

1073

d) SHALL only contain requirements that are testable.

1074

1075

1076 A conforming implementation SHALL satisfy the conformance requirements of the
1077 applicable parts of this specification and the appropriate technical specification(s).

1078

1079 **10.4 General Framework of Conformance Testing**

1080

1081 The objective of conformance testing is to determine whether an implementation being
1082 tested conforms to the requirements stated in the relative ebXML specification.

1083

1084 Conformance testing enables vendors to implement compatible and interoperable systems
1085 built on the ebXML foundations. EbXML *implementations* and *Applications* SHALL be
1086 tested to available test suites to verify their conformance to ebXML Specifications.

1087

1088

1089 Publicly available test suites from vendor neutral organizations such as OASIS and NIST
1090 SHOULD be used to verify the conformance of ebXML *Implementations*, *Applications*,
1091 and *Components* claiming conformance to ebXML. Open source reference
1092 implementations MAY be available to allow vendors to test their products for interface
1093 compatibility, conformance, and interoperability.

1094

1095

1096

1097

1098

1099

1100 **11.0 Security Considerations**

1101

1102 **11.1 Introduction**

1103

1104 A comprehensive *Security Model* for ebXML will be expressed in a separate document.

1105

1106 The *Security Model* SHALL be applied to the entire ebXML Infrastructure, with the
1107 underlying goal of best meeting the needs of users of ebXML.

1108

1109

1110

1111 The Security Model SHALL comply with security needs specified in the *ebXML*
1112 *Requirements Document*.

1113

1114

1115

1102 **Appendix A: Example ebXML Business Scenarios**

1103 **Definition**

1104 This set of Scenarios defines how ebXML compliant software could be used to
1105 implement popular, well-known *eBusiness* models.

1106 **Scope**

1107 These Scenarios are oriented to properly position ebXML specifications as a convenient
1108 mean for SME's to properly conduct *eBusiness* over the Internet using open standards.
1109 They bridge the specifications to real life uses.

1110 **Audience**

1111 Companies planning to use ebXML compliant software will benefit from these scenarios
1112 because they will show how these companies MAY be able to implement popular
1113 business scenarios onto the ebXML specifications.

1114 **List**

- 1115 a) Two *Trading Partners* set-up an agreement and run the associated electronic
1116 exchange.
- 1117 b) Three or more *Trading Partners* set-up a *Business Process* implementing a
1118 supply-chain and run the associated exchanges
- 1119 c) A company sets up a Portal which defines a *Business Process* involving the use of
1120 external business services.
- 1121 d) Three or more *Trading Partners* engage in multi-Party *Business Process* and run
1122 the associated exchanges.

1123

1124 **Scenario 1**

1125 **Two *Trading Partners* set-up an agreement and run the associated exchange.**

1126 In this scenario:

- 1127 • Each *Trading Partner* defines its own *Trading Partner Profile (TPP)*. Each
1128 *TPP* references:
 - 1129 • One or more existing *Business Process* found in an ebXML *Registry*
1130 system.
 - 1131 • One of more *Business Message* definitions. Each definition is built from
1132 reusable *Components (Core and/or Aggregate Components)* found in the
1133 ebXML *Registry* system.

1134

1135 Each *TPP* defines:

- 1136 • The commercial transactions that the *Trading Partner* is able to support.
- 1137 • The underlying protocol (like HTTP, SMTP etc) and the technical properties
1138 (such as encryption, validation, authentication, digital signing) that the
1139 *Trading Partner* supports in the engagement.
- 1140 • The *Trading Partners* acknowledge each other's *TPP* and create a
1141 *Collaboration Protocol Agreement (CPA)*. The *CPA* references:
 - 1142 • The relevant *TPP*'s.
 - 1143 • The Legal terms and conditions related to the exchange
 - 1144 • The parties implement the respective part of the Profile. This is done:
 - 1145 • Either by creating/configuring a *Business Service Interface*.

- 1146 • Or properly upgrading the legacy software running at their side
- 1147 • In both cases, this step is about:
 - 1148 • Plugging the legacy into the ebXML technical
 - 1149 infrastructure as specified by the *ebXML Messaging*
 - 1150 *Services Specification*.
 - 1151 • Granting that the software is able to properly engage the
 - 1152 stated conversations
 - 1153 • Granting that the exchanges semantically conform to the
 - 1154 agreed upon message definitions
 - 1155 • Granting that the exchanges technically conform with the
 - 1156 underlying *ebXML Messaging Service*.
 - 1157 • The *Trading Partners* start exchanging messages and
 - 1158 performing the agreed upon commercial transactions.

1159
1160 **Scenario 2:**

1161 **Three or more *Trading Partners* set-up a *Business Process* implementing a supply-chain**
1162 ***eBusiness* scenario.**

1163
1164 The simple case of a supply-chain involving two *Trading Partners* can be reconstructed
1165 from the Scenario 1.

1166 Here we are dealing with situations where more parties are involved. We consider a
1167 *Supply Chain* of the following type:



1168
1169
1170
1171
1172
1173 What fundamentally differs from Scenario 1 is that *Trading Partner 2* is engaged at the
1174 same time with two different *Trading Partners*. The assumption is that the “state” of the
1175 entire *Business Process* is managed by each *Trading Partner*, i.e. that each *Trading*
1176 *Partner* is fully responsible of the commercial transaction involving it (*Trading Partner 3*
1177 only knows about *Trading Partner 2*, *Trading Partner 2* knows about *Trading Partner 3*
1178 and *Trading Partner 1*, *Trading Partner 1* knows about *Trading Partner 2*).

1179
1180 In this scenario:

- 1181 • Each *Trading Partner* defines its’ own *TPP*. Each *TPP* references:
 - 1182 • One or more existing *Business Process* found in the ebXML *Registry* system.
 - 1183 • One of more *Business Message* definitions. Each definition is built from
 - 1184 reusable *Components (Core and/or Aggregate Components)* found in the
 - 1185 ebXML *Registry*.
- 1186 • Each *TPP* defines:
 - 1187 • The commercial transactions that the *Trading Partner* is able to engage
 - 1188 into
 - 1189 *Trading Partner 2* must be able to support at least 2 commercial
 - 1190 transactions.

- 1191
- 1192
- 1193
- 1194
- 1195
- 1196
- 1197
- 1198
- 1199
- The underlying protocol (like HTTP, SMTP etc) and the technical properties (such as encryption, validation, authentication, and digital signing) that the *Trading Partner* supports in the engagement.
 - The technical requirements for the exchanges with *Trading Partner 1* and *Trading Partner 3* MAY be different. In such case, *Trading Partner 2* must be able to support different protocols and/or properties.
 - The *Trading Partners* acknowledge each other *TPP* and create the relevant *CPA*'s (at least 2 in this scenario).
- 1200
- Each *CPA* references:
 - The relevant *CPP*'s for each respective *Trading Partner*.
 - The terms and conditions related to the mutually agreed upon business exchange
 - *Trading Partner 2* is engaged in 2 *CPA*'s. Each *Trading Partner* implements their own respective part of each *CPA*. This is done:
 - Either by creating/configuring a *Business Service Interface*.
 - Or properly upgrading the legacy software running at their side . In both cases, this step is about:
 - Plugging the Legacy into the ebXML technical infrastructure as specified by the TR&P
 - Granting that the software is able to properly engage the stated conversations
 - Granting that the exchanges semantically conform to the agreed upon *Business Message* definitions
 - Granting that the exchanges technically conform with the underlying ebXML Messaging Service.
 - *Trading Partner 2* MAY need to implement a complex *Business Service Interface* in order to be able to engage with different *Trading Partners*.
 - The *Trading Partners* start exchanging messages and perform the agreed upon commercial transactions.
 - *Trading Partner 3* places an order with *Trading Partner 2*.
 - *Trading Partner 2* (eventually) places an order with *Trading Partner 1*.
 - *Trading Partner 1* fulfills the order.
 - *Trading Partner 2* fulfills the order.
- 1201
- 1202
- 1203
- 1204
- 1205
- 1206
- 1207
- 1208
- 1209
- 1210
- 1211
- 1212
- 1213
- 1214
- 1215
- 1216
- 1217
- 1218
- 1219
- 1220
- 1221
- 1222
- 1223
- 1224
- 1225

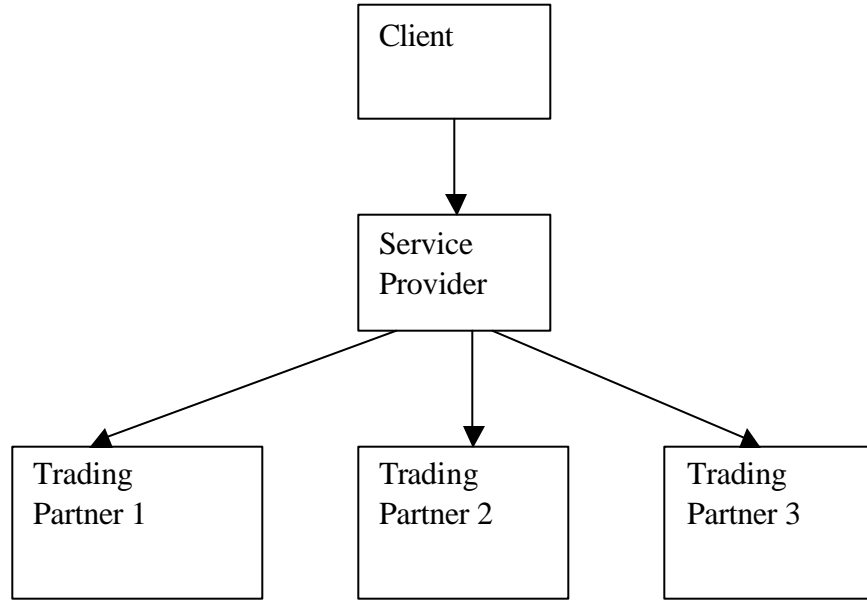
1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1270

Scenario 3

A Company sets up a Portal which defines a Business Process involving the use of external business services

This is the scenario describing a *Service Provider*. A “client” asks the *Service Provider* for a service. The *Service Provider* fulfills the request by properly managing the exchanges with other partners, which provide information to build the final answer.

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

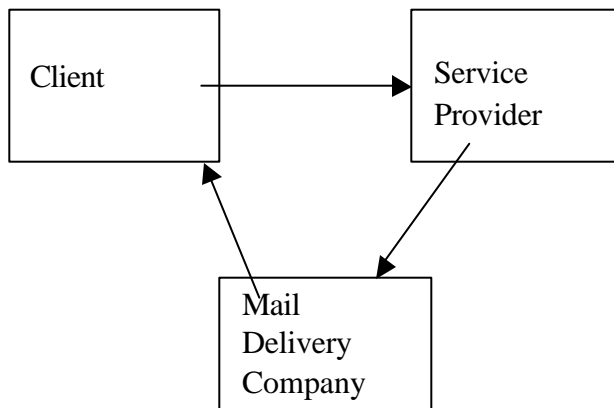


This is an evolution of Scenario 2. The Description of this scenario is omitted to minimize the duplication of processes explained in detail in Scenario 2.

Scenario 4

Three or more Trading Partners engage in eBusiness using Business Processes that were created by each respective Trading Partner and run the associated business exchanges.

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



1271

1272 In this Scenario, each *Trading Partner* is involved with more than one other *Trading*
1273 *Partner* but the relationship is not linear. The product ordered by the client from the
1274 Service Provider will be delivered by a 3rd *Trading Partner*.

1275

1276 In this scenario:

- 1277 • Each *Trading Partner* defines its own *CPP*.
1278 Each *CPP* references:
 - 1279 • One or more existing *Business Process* found in the ebXML *Registry*
 - 1280 • One of more *Business Message* definitions. Each definition is built from reusable
1281 *Components (Core and/or Aggregate Components)* found in the ebXML *Registry*

1282

1283 Each *CPP* defines:

- 1284 • The Commercial Transactions that the *Trading Partner* is able to engage into
1285 In this case, each *Trading Partner* must be able to support at least 2 commercial
1286 transactions.
 - 1287 • The technical protocol (like HTTP, SMTP etc) and the technical properties (such
1288 as encryption, validation, authentication, and digital signing) that the *Trading*
1289 *Partner* supports in the engagement.
1290 In case the technical infrastructure underlying the different exchanges differs,
1291 each *Trading Partner* must be able to support different protocols and/or
1292 properties. (an example is that the order is done through a Web site and the
1293 delivery is under the form of an email).
 - 1294 • The *Trading Partners* acknowledge each other profile and create a *Partner CPA*.
1295 Each *Trading Partner*, in this Scenario, must be able to negotiate at least 2 *CPA*'s.
 - 1296 • The *CPA* references:
 - 1297 • The relevant *CPP*'s
 - 1298 • The terms and conditions related to the exchange
- 1299 • Each *Trading Partner* is engaged in 2 *CPA*'s.
 - 1300 • The *Trading Partners* implement the respective part of the Profile. This is
1301 done:
 - 1302 • Either by creating/configuring a *Business Service Interface*.
 - 1303 • Or properly upgrading the legacy software running at their side
- 1304 • In both cases, this step is about:
 - 1305 • Plugging the application into the ebXML technical infrastructure as
1306 specified by the *ebXML Messaging Service*.
 - 1307 • Granting that the software is able to properly engage the stated
1308 business scenarios.
 - 1309 • Granting that the exchanges semantically conform to the agreed upon
1310 *Business Message* definitions
 - 1311 • Granting that the exchanges technically conform with the underlying
1312 *ebXML Messaging Services Specification*.

1314

- 1315 • All *Trading Partners* MAY need to implement complex *Business*
- 1316 *Service Interfaces* to accommodate the differences in the *CPA's* with
- 1317 different *Trading Partners*.
- 1318 • The *Trading Partners* start exchanging messages and performing the
- 1319 agreed upon commercial business transactions.
- 1320 • The Client places an Order at the Service Provider
- 1321 • The Service Provider acknowledges the Order with the Client
- 1322 • The Service Provider informs the mail delivery service about a product
- 1323 to be delivered at the Client
- 1324 • The Mail Delivery Service delivers the product to the Client
- 1325 • The Clients notifies the Service Provider that the product is received.

1326 ***Disclaimer***

1327 The views and specification expressed in this document are those of the authors and are
1328 not necessarily those of their employers. The authors and their employers specifically
1329 disclaim responsibility for any problems arising from correct or incorrect implementation
1330 or use of this design.

1331 ***Copyright Statement***

1332 Copyright © ebXML 2000. All Rights Reserved.

1333

1334 This document and translations of it MAY be copied and furnished to others, and
1335 derivative works that comment on or otherwise explain it or assist in its implementation
1336 MAY be prepared, copied, published and distributed, in whole or in part, without
1337 restriction of any kind, provided that the above copyright notice and this paragraph are
1338 included on all such copies and derivative works. However, this document itself MAY
1339 not be modified in any way, such as by removing the copyright notice or references to the
1340 Internet Society or other Internet organizations, except as needed for the purpose of
1341 developing Internet standards in which case the procedures for copyrights defined in the
1342 Internet Standards process must be followed, or as REQUIRED to translate it into
1343 languages other than English.

1344

1345 The limited permissions granted above are perpetual and will not be revoked by ebXML
1346 or its successors or assigns.

1347

1348 This document and the information contained herein is provided on an
1349 "AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR
1350 IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE
1351 USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1352 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1353 PARTICULAR PURPOSE.