



Creating A Single Global Electronic Market

1

2 **ebXML Technical Architecture Specification**

3 **ebXML Technical Architecture Team**

4

5 17 October 2000

6 ***1.0 Status of this Document***

7

8 This document represents a work in progress upon which no reliance should be made.
9 Distribution of this document is unlimited. The document formatting is based on the
10 Internet Society's Standard RFC format.

11

12 ***This version:***

13 ebXML_TA_v0.9.doc

14

15 ***Latest version:***

16 ebXML_TA_v0.9.doc

17

18 ***Previous version:***

19 EbXML_TA_v0.8.72i

20

21 ***2.0 Scope***

22

23 This document describes the underlying *Architecture* for ebXML. It provides a high level
24 overview of ebXML and describes the relationships, interactions, and basic functionality
25 of ebXML *Components*. It should be used as a roadmap to learn: (1) what ebXML is, (2)
26 what problems ebXML solves, and (3) core ebXML functionality. This document does
27 not go into the level of detail required to build an ebXML *Application*. Please refer to
28 each of the ebXML component specifications for the exact information needed to build
29 ebXML *Applications* and related *Components*.

30

31 ***3.0 Normative References***

32

33 The following standards contain provisions which, through reference in this text,
34 constitute provisions of this specification. At the time of publication, the editions
35 indicated below were valid. All standards are subject to revision, and parties to
36 agreements based on this specification are encouraged to investigate the possibility of
37 applying the most recent editions of the standards indicated below.

38

39 W3C XML v1.0 Specification

- 40 ISO/IEC 14662: Open-edi Reference Model
- 41 ISO 11179 Metadata Repository
- 42 ISO 10646: Character Encoding
- 43 ISO 8601:2000 Date/Time/Number Data typing
- 44

45 ***4.0 ebXML Technical Architecture Participants***

46
 47 We would like to recognize the following for their significant participation in the
 48 development of this document.

49
 50 Editors: Duane Nickull, XML Global Technologies
 51 Brian Eisenberg, DataChannel

52
 53 Participants: Colin Barham, TIE
 54 Al Boseman
 55 Dick Brooks, Group 8760
 56 Cory Casanave, DataAccess Technologies
 57 Robert Cunningham, Military Traffic Management Command, US Army
 58 Christopher Ferris, Sun Microsystems
 59 Anders Grangard, EDIFrance
 60 Kris Ketels, SWIFT
 61 Piming Kuo, Worldspan
 62 Kyu-Chul Lee, Chungnam National University
 63 Henry Lowe, OMG
 64 Melanie McCarthy, General Motors
 65 Klaus-Dieter Naujok, NextEra Interactive
 66 Bruce Peat, eProcessSolutions
 67 John Petit, KPMG Consulting
 68 Mark Heller, MITRE
 69 Scott Hinkelman, IBM
 70 Karsten Riemer, Sun Microsystems
 71 Lynne Rosenthal, NIST
 72 Nikola Stojanovic, Columbine JDS Systems
 73 Jeff Sutor, Sun Microsystems
 74 David RR Webber, XML Global Technologies

75
 76

76 **5.0 Table of Contents**

77 ebXML Technical Architecture Specification 1

78 1.0 Status of this Document 1

79 2.0 Scope 1

80 3.0 Normative References 1

81 4.0 ebXML Technical Architecture Participants..... 2

82 5.0 Table of Contents 3

83 6.0 Introduction 5

84 7.0 ebXML Abstract Overview 6

85 8.0 ebXML Conceptual Overview 7

86 9.0 Relating the ebXML Architecture to Existing Standards..... 9

87 10.0 ebXML Architecture 11

88 11.0 ebXML Business Operational View 12

89 12.0 ebXML Functional Service View 14

90 13.0 Implementation Phase 17

91 14.0 Discovery and Deployment Phase..... 18

92 15.0 Run Time Phase 19

93 16.0 Trading Partner Information..... 19

94 16.1 Support for Trading Partner Agreements 20

95 17.0 Business Process and Information Modeling 21

96 17.1 Overview 21

97 17.2 Position within overall ebXML Architecture..... 21

98 17.3 Business Process and Information Modeling Functionality..... 22

99 17.4 The Business Process and Information Metamodel 22

100 17.5 Interfaces and Relationship 23

101 17.6 Relationship to Trading Partner Agreements 24

102 17.7 Relationship to Core Components..... 24

103 18.0 Core Component Functionality 24

104 19.0 Business Object Functionality..... 25

105 19.1 Overview 25

106 19.2 Business Objects in Business Process and Information Models 26

107 19.3 Common Business Objects..... 26

108 20.0 Registry and Repository Functionality..... 27

109 20.1 Overview 27

110 20.2 Information Model & Interface Constrains 28

111 20.3 Formal Functional Overview 29

112 20.4 Sample Objects Residing in a Repository and Managed by a Registry 30

113 20.5 Registry Management of Repository Objects and Metadata..... 30

114 20.6 Querying Registries and Returning Repository Objects and Metadata..... 31

115 20.7 Registry to Registry Interfacing Model..... 31

116 20.8 Registry/Repository Business Scenario Example 33

117 21.0 Messaging Service Functionality 33

118 21.1 Overview 33

119 21.2 Abstract ebXML Messaging Service Interface 34

120 21.3 ebXML Messaging Service Layer Functions..... 34

121 21.4 ebXML Message Structure and Packaging 37

122 22.0 Conformance 38

123 22.1 Overview 38

124 22.2 Conformance Requirements..... 38

125 22.3 General Framework of Conformance Testing..... 39

126 Scenario 1: Two Partners set-up an agreement and run the associated exchange..... 40

127 Scenario 2: Three or more partners set-up a Business Process implementing a

128 supply-chain and run the associated exchanges 42

129 Scenario 3 : A Company sets up a Portal which defines a Business Process involving

130 the use of external business services 43

131 Scenario 4: Three or more parties engage in multi-party Business Process and run

132 the associated exchanges 44

133 Disclaimer 46

134 Copyright Statement..... 46

135

135 **6.0 Introduction**

136

137 Over 25 years ago the idea was born to eliminate the use of paper documents for
138 exchanging business data by linking computer systems together so that the data, normally
139 on paper, could be sent from one system to the other. This concept became known as
140 *Electronic Data Interchange (EDI)*. The advantages are still valid today: single point of
141 information capture, electronic delivery, low storage and retrieval costs, to mention just a
142 few. However, looking at the statistics of who is currently utilizing *EDI* only the top
143 10,000 companies on a global scale (Fortune 1000 in the top 10 countries) are using *EDI*.
144 For the rest of the business world only 5% are using *EDI* and therefore today common
145 *Business Processes* are dominated by paper transactions.

146

147 Today, *Extensible Markup Language (XML)* is at the forefront of efforts to replace paper-
148 based business transactions. In order for *Small to Medium Enterprises (SMEs)* to benefit
149 from the next generation of *eBusiness* standards, these standards must contain all the
150 information to allow software developers to create programs that can be purchased off-
151 the-self (shrink-wrapped-solutions) or developed in-house. The success of any new way
152 to exchange data among businesses depends not only on the adoption by the Fortune
153 1000 companies of standard agreements, but on their adoption by the other estimated
154 25,000,000 *SMEs* in the world. Without an economic incentive for the *SMEs*, any new
155 method of accomplishing *eBusiness* is just re-inventing the status quo instead of
156 delivering a pervasive solution.

157

158 The answer is to document and capture in an unambiguous way the *Business Processes*
159 and associated information requirements for a particular business goal, which can then be
160 processed by a computer program. The use of *XML* technologies combined with
161 *Business Process and Information Modeling* and object-oriented technology can achieve
162 this objective. Instead of looking at the data requirements based on internal legacy
163 database records, *Business Experts* identify the collaborations with other parties in order
164 to achieve a certain business goal. Those collaborations are documented in a model
165 developed in the *Unified Modeling Language (UML)*. Each activity requires the
166 exchange of business information. Instead of taking the *Data Element (EDI)* approach,
167 *Objects* are used to describe and model *Business Processes*.

168

169 With the advent of *XML*, it is easier to identify and define *Objects* with attributes (data)
170 along with functions that can be performed on those attributes. There are many *Objects*
171 that are common to many *Business Processes* (goals), such as address, *Party*, and
172 location.. By allowing these *Objects* to be reused, ebXML can provide the means to
173 unify cross-industry exchanges with a single consistent *Lexicon*. However the role of
174 ebXML is not to replicate the reliance on electronic versions of common paper
175 documents such as purchase orders, invoices and tender requests and to offer up and
176 develop such implementation examples. Instead the ebXML specifications provide a
177 framework where *SMEs*, software engineers, and other organizations can create
178 consistent, robust, and interoperable *eBusiness* services and *Components*, ultimately
179 leading to the realization of global *eBusiness*.

180

181 **7.0 ebXML Abstract Overview**

182

183 Although *XML* is a recent newcomer in the *eBusiness* landscape, *Supply Chains* in many
184 industries, as well as industry consortiums and standards organizations are using *XML* to
185 define their own vocabularies for business relationships and transactions. The
186 vocabularies, business templates, and *Business Processes* used by these groups to transact
187 business must be accessible by all partners at any time.

188

189 Furthermore, newcomers to the *Supply Chain* or business partnerships must be able to
190 discover and implement *eBusiness* interfaces to interoperate in a secure, reliable and
191 consistent manner. In order to facilitate these needs, mechanisms must be in place that
192 can provide information about each participant (*Trading Partner*), including what they
193 support for *Business Processes* and their implemented service interfaces. This includes
194 information about what business information is required for each instance of a business
195 message, and a mechanism to allow dynamic discovery of the semantic meaning of that
196 business information. The entire mechanism must be able to recognize semantic
197 meanings at the business element level and be implemented using *XML*-based
198 representations and systems. The complete set of ebXML Specifications explains this
199 functionality in detail.

200

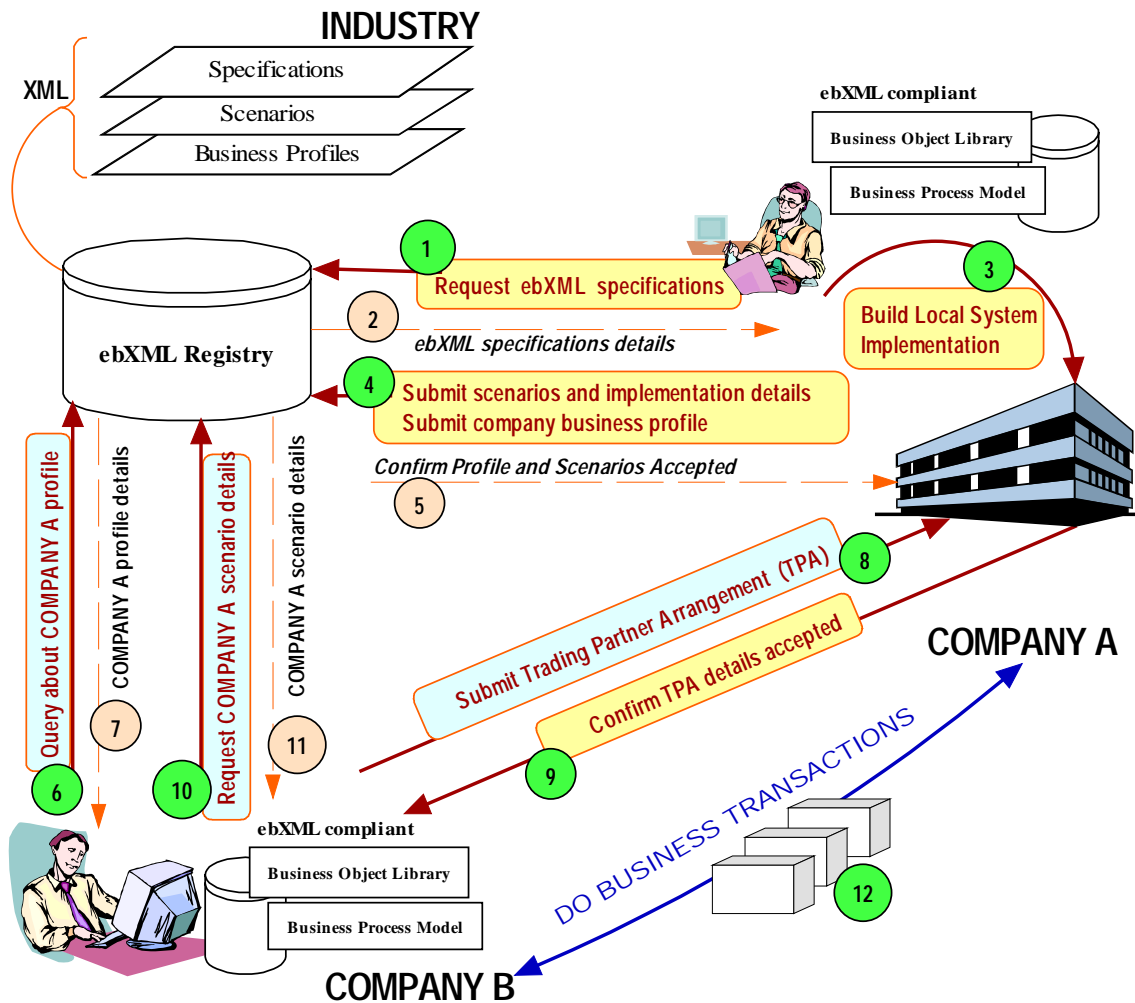
200

201 8.0 ebXML Conceptual Overview

202

203 Figure 1 shows a conceptual model for two *Trading Partners*, first configuring and then
 204 engaging in a simple business transaction interchange. This model is provided as an
 205 illustration of the process and steps that may typically be required using ebXML
 206 *Applications* and related *Components*. The ebXML specifications are not limited to this
 207 simple model, provided here as quick introduction to the concepts. Further examples of
 208 ebXML implementation models are provided at the end of this section. Specific
 209 implementation examples are described in Appendix A.

210



211
212
213
214

Figure 1: a high level overview of ebXML functionality

215 In Figure 1, Company A has become aware of an *ebXML Registry* that contains a set of
 216 ebXML Specifications. Company A requests an ebXML Specification in order to
 217 determine if it wants to become an ebXML compliant participant (Figure 1, step 1). The

218 request results in the ebXML process specification being sent to Company A (Figure 1,
219 step 2). Company A, after reviewing the specification, decides to build and deploy its
220 own ebXML compliant application (Figure 1, step 3). [Note: custom software
221 development is not a necessary prerequisite for ebXML participation, user *Applications*
222 will be also commercially available as turn-key solutions.]
223

224 Company A then submits its own implementation details, reference links, and *Trading*
225 *Partner Profile (TPP)* as a request to the *ebXML Registry* (Figure 1, step 4). The *TPP*
226 submitted describes the company's ebXML capabilities and constraints, as well as its
227 supported business scenarios. These scenarios are *XML* versions of the *Business*
228 *Processes* and associated information parcels (based on *Business Objects*: for example a
229 sales tax calculation) that the company is able to engage in. After receiving verification
230 that the format and usage of a *Business Object* is correct, an acknowledgment is sent to
231 Company A by the *ebXML Registry* (Figure 1, step 5).
232

233 Company B (an SME) is then informed by Company A that they would like to engage in
234 a business transaction using ebXML. Company B acquires a shrink-wrapped application
235 that is ebXML compliant and able to interface with its existing (legacy) *Applications*. The
236 ebXML program already contains the base ebXML information bundles such as a library
237 of *Business Objects* and Models for the specific industry they are part of. Company A
238 knows that its *Business Processes* and *TPP* are compliant with the ebXML infrastructure
239 from the information available in the ebXML specification package. However, since
240 Company A just registered its scenarios, they are not yet part of the package. Therefore
241 the ebXML *Application* queries the *ebXML Registry* about Company A (Figure 1, step 6).
242 Company A's profile is retrieved (Figure 1, step 7). Based on the *TPP*, the *Application*
243 determines that it is able to execute a specific scenario that Company A supports.
244

245 Before engaging in that the scenario Company B submits a proposed *Trading Partner*
246 *Agreement (TPA)* directly to Company A's ebXML compliant software interface. The
247 *TPA* outlines the *eBusiness* scenario and specific arrangement(s) it wants to use with
248 Company A, as well as certain messaging, contingency and security-related requirements
249 (Figure 1, step 8). Company A accepts the *TPA* and acknowledgement is sent directly to
250 Company B's shrink-wrapped ebXML software *Application* (Figure 1, step 9). Since the
251 scenario from Company A was not available in the software package that Company B is
252 using, the *Application* requests it from the *ebXML Registry* (Figure 1, step 10). The
253 scenario is then provided to Company B's *Application* (Figure 1, step 11).
254

255 Based on the processes (contained in the process models) and information parcels
256 (presented in *Class Diagrams*) Company A and B are now engaging in *eBusiness*
257 utilizing ebXML specifications via their respective software *Applications* (Figure 1, step
258 12).
259

259 The conceptual overview described in the scenario above introduced the following
260 concepts and architectural *Components*:

261

- 262 1. A standard mechanism for describing a *Business Process* and its associated
263 information model.
- 264 2. A mechanism for registering and storing a *Business Process and Information*
265 *Metamodel* so that it can be shared/reused.
- 266 3. Discovery of information about each participant including:
 - 267 • The *Business Processes* they support.
 - 268 • The *Business Service Interfaces* they offer in support of the *Business Process*.
 - 269 • the *Business Messages* are to be exchanged between their respective *Service*
270 *Interfaces*.
 - 271 • The technical configuration of the supported transport, security and encoding
272 *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 a *Trading Partner Agreement (TPA)* which may be
276 derived from the information about each participant from item 3 above.
- 277 6. A standardized *Messaging Service* which enables interoperable, secure and
278 reliable exchange of messages between two parties.
- 279 7. A mechanism for configuration of the respective *Messaging Services* to engage in
280 the agreed upon *Business Process* in accordance with the constraints defined in
281 the *TPA*.

282

283 Using these *Components* ebXML compliant software can be used to implement popular,
284 well-known *eBusiness* scenarios, examples include but are not limited to:

285

- 286 a) Two partners set-up an agreement and run the associated electronic exchange.
- 287 b) Three or more partners set-up a *Business Process* implementing a supply-chain
288 and run the associated electronic exchanges
- 289 c) A company sets up a portal that defines a *Business Process* involving the use of
290 external business services.
- 291 d) Three or more parties engage in multi-Party *Business Process* and run the
292 associated electronic exchanges.

293

294 The above examples are described in detail in Appendix A.

295 **9.0 Relating the ebXML Architecture to Existing Standards**

296

297 The ebXML approach utilizes public specifications and standards wherever applicable
298 and consistent with the goals of the ebXML initiative. One such specification is the
299 *Open-edi* work, an ISO/IEC 14662 (*Open-edi Reference Model*) vision of future *EDI*.
300 The ebXML approach can benefit from the lessons learned by *Open-edi* work and utilize
301 the related methodologies. Particularly, *Open-edi* takes a generic industry and technology
302 neutral approach and by similarly utilizing this, ebXML will enable organizations to

303 provide the opportunity to significantly lower the barriers to electronic data exchange by
 304 introducing standard business scenarios and the necessary services to support them. In
 305 principle, once a business scenario is agreed upon, and implementations conform to the
 306 standards, there is no need for prior agreement among *Trading Partners*, other than the
 307 decision to engage in the ebXML transaction in compliance with the business scenario.
 308 This will lead to the ability to establish short-term business relationships quickly and cost
 309 effectively.

310

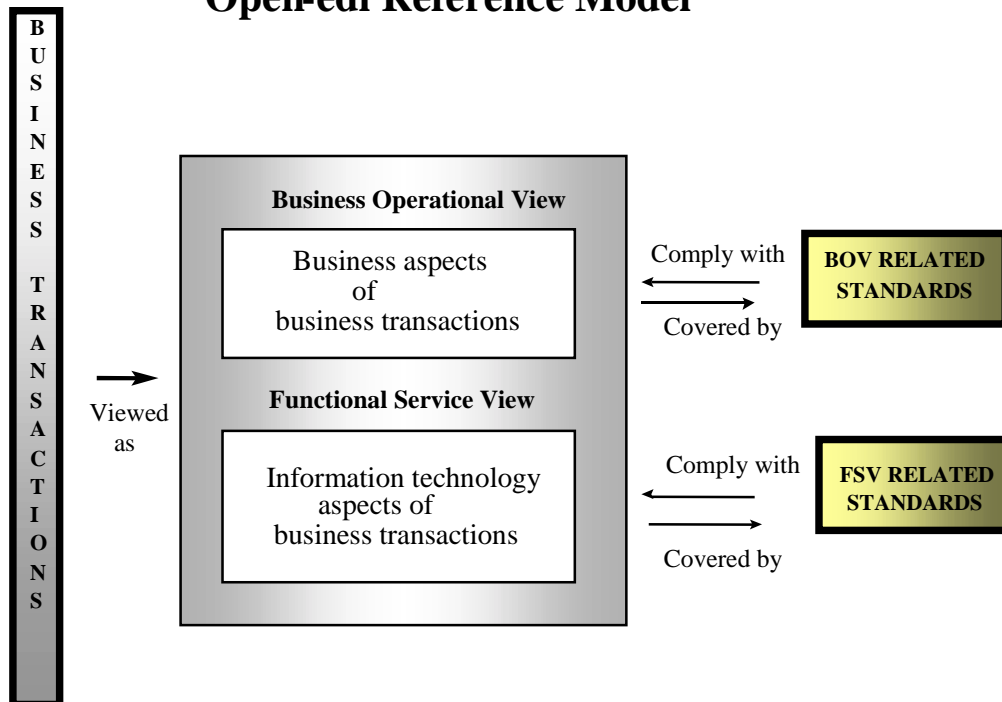
311 The field of *Application* of ebXML is the electronic processing of *XML*-based business
 312 transactions among autonomous multiple organizations within and across sectors (e.g.,
 313 public, private, industrial, geographic). It includes business transactions that involve
 314 multiple data types such as numbers, characters, images and sound. The *Open-edi*
 315 *Reference Model* provides the standards required for the inter-working of organizations
 316 through interconnected *Information Technology* systems, and is independent of specific
 317 *Information Technology* (IT) implementations, business content or conventions, business
 318 activities, and organizations.

319

320 The *Open-edi Reference Model* places existing *EDI* standards in perspective using two
 321 views to describe the relevant aspects of business transactions: the *Business Operational*
 322 *View (BOV)* and the *Functional Service View (FSV)*. The ebXML Architecture uses
 323 similar views of these definitions. The *BOV* expresses the users' requirements needed to
 324 achieve the common business goal. The *FSV* describes how the *BOV* is actually
 325 implemented using the selected technology.

326

Open-edi Reference Model



327

328

Figure 2: Open-edi environment

329

330 Figure 2 above sets out the relationship between the *Open-edi Reference Model* and these
331 views.

332 The primary focus of ebXML resides with the *FSV* and the supporting *BOV*. The
333 assumption for ebXML is that the *FSV* will be implemented by commercial software
334 vendors and ensure backwards compatibility to traditional *EDI* systems (where
335 applicable). As such, the resultant *BOV*-related standards provide the business and object
336 class models needed to construct ebXML compliant *eBusiness* services and *Components*.
337

338 While business practices from one business organization to another are highly variable,
339 most activities can be decomposed into *Business Processes* which are more generic to a
340 specific type of business. This analysis through the modeling process will identify object
341 classes and models that are likely candidates for standardization. The ebXML approach
342 looks for standard reusable *Components* from which to construct information exchange
343 software. While *Open-edi* is a theoretical syntax neutral approach, ebXML itself is
344 focused on a physical implementation using specifically an XML-based syntax and
345 related technologies.
346

347 **10.0 ebXML Architecture**

348

349 The ebXML Architecture Reference Model uses the following two views to describe the
350 relevant aspects of business transactions:

351

- 352 • The Business Operational View (*BOV*)
- 353 • The Functional Service View (*FSV*)

354

355 The *BOV* addresses the *Semantics* of:

356

357 a) The *Semantics* of business data in transactions and associated data interchanges

358

359 b) The architecture for business transactions, including:

360

- 361 ○ operational conventions;
- 362 ○ agreements;
- 363 ○ mutual obligations and requirements.

364

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

366

367 The *FSV* addresses the supporting services meeting the mechanistic needs of ebXML. It
368 focuses on the *Information Technology* aspects of:

369

- 370 • functional capabilities;
- 371 • *Service Interfaces*;

- *Protocols.*

372

373

374 Additionally, the functional capabilities, *Service Interfaces* and *Protocols* include:

375

376

- capabilities for implementation, discovery, deployment and run time scenarios;
- user *Application* interfaces;
- data transfer infrastructure interfaces;
- *Protocols* for interworking of XML vocabulary deployments from different organizations.

377

378

379

380

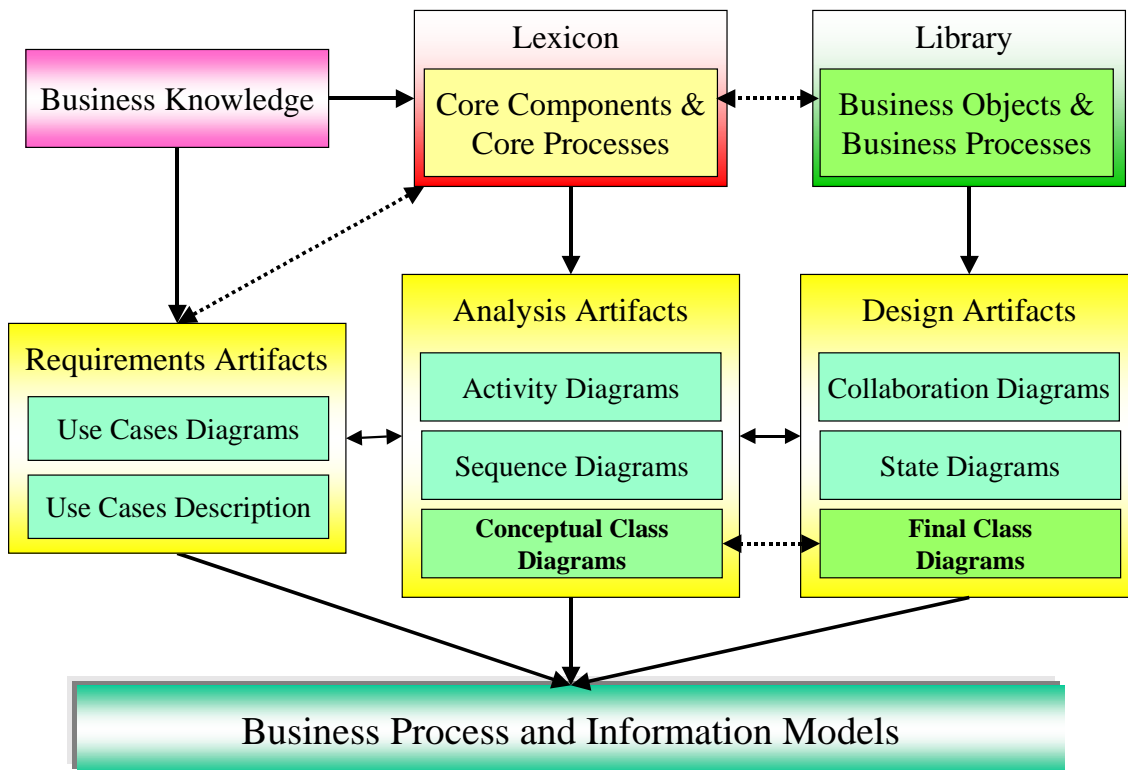
381

382 The ebXML *BOV* and the *FSV* are discussed in detail in the following sections.

383

384 **11.0 ebXML Business Operational View**

385



386

387

388

389

Figure 3: the Business Operational View

390

390 *ebXML Business and Information Models* are created following the selected ebXML
391 *Business Process and Information Modeling* (see section 17).

392

393

393 *Business Knowledge* is captured in a *Lexicon*. The *Lexicon* contains data and process
394 definitions including relationships and cross-references as expressed in business
395 terminology and organized by industry domain. The *Lexicon* is the bridge between the

396 specific business or industry language and the knowledge expressed by the models in a
 397 more generalized industry neutral language.

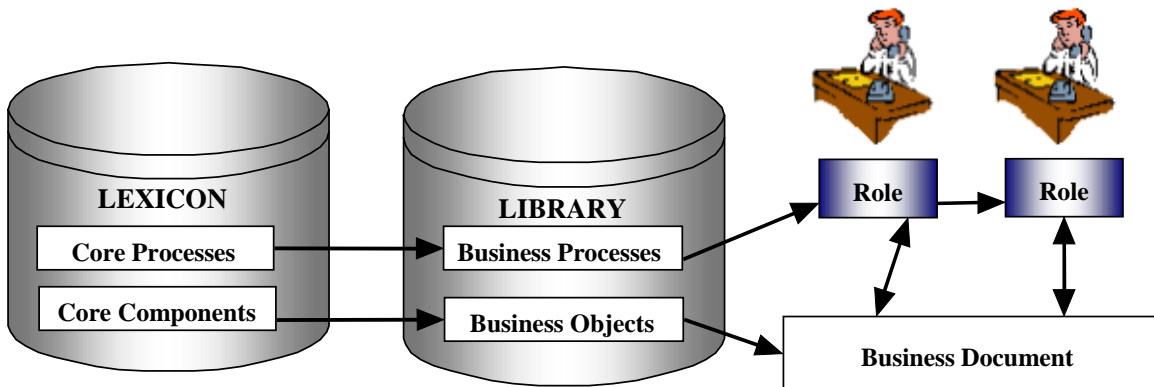
398
 399 The first phase defines the requirements artifacts which describe the problem using *Use*
 400 *Case Diagrams and Descriptions*. If *Lexicon* entries are available they will be utilized,
 401 otherwise new *Lexicon* entries will be created.

402
 403 The second phase (analysis) will create activity and sequence diagrams describing the
 404 *Business Processes*. *Class Diagrams* will capture the associated information parcels
 405 (business messages). The analysis phase reflects the business knowledge contained in the
 406 *Lexicon*. No effort is made to force the *Application* of object-oriented principles. The
 407 class diagram is a free structured data diagram.

408
 409 The design phase is the last step of standardization, which may be accomplished by
 410 applying object-oriented principles. In addition to generating collaboration diagrams, a
 411 state diagram may also be created. The data diagram from the analysis phase will
 412 undergo harmonization to align it with other models in the same industry and across
 413 others.

414
 415 Therefore in ebXML interoperability is achieved by applying *Business Objects* across all
 416 class models. The content of the *Business Object Library* is created by analyzing existing
 417 *Business Objects* as used by many industries today in conjunction with the *Lexicon*
 418 content and ebXML selected modeling methodology.

419
 420 Figure 4 shows how the user can see this correlation to the actual business roles:
 421



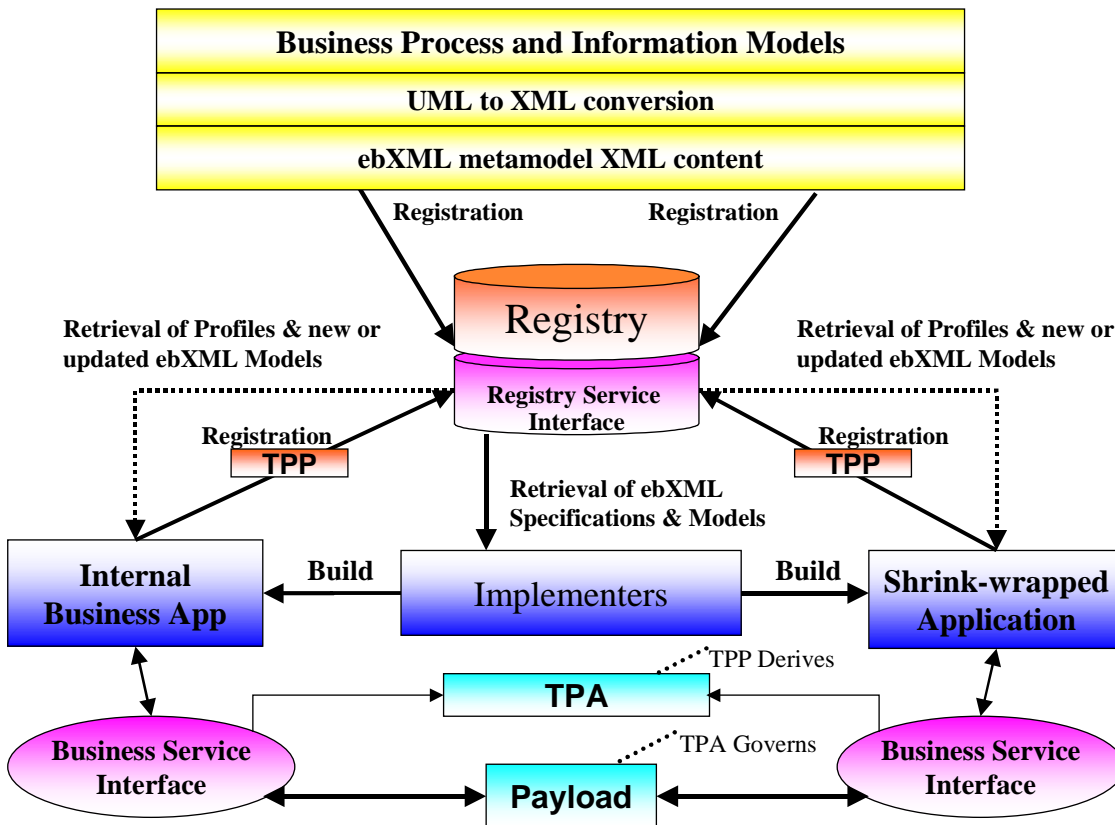
422
 423
 424

Figure 4: Role Relation Model.

425

426 **12.0 ebXML Functional Service View**

427



428

429

430

431

Figure 5: ebXML Functional Service View

432 The *ebXML Registry* system is an important part of ebXML. The *Registry* not only holds
 433 the ebXML base reference specifications, but also the *Business Process and Information*
 434 *Metamodels* developed by industry groups, *SMEs*, and other organizations. These models
 435 are compliant with the *ebXML Metamodel* and related methodologies. In order to store
 436 the models they are converted from *UML* to *XML*. ebXML Registries store these models
 437 as instances of *XML* that are compliant to the *ebXML Metamodel*.

438

439 This *XML*-based business information shall be expressed in a manner that will allow
 440 discovery down to the attribute level via a consistent methodology. In order to enable this
 441 functionality, the use of *Unique Identifiers (UIDs)* is required for all items within an
 442 *ebXML Registry* and *Repository System*. These *UID* references are implemented as *XML*
 443 attributes, expressed as fixed value attributes for each of the physical *XML* elements and
 444 structures. *UID* keys are required references for all ebXML content. The *UID* keys
 445 themselves do not contain explicit versioning control, but may be used with versioning
 446 control mechanisms, either as an extension to the *UID* key value itself, or within the

447 *ebXML Registry and Repository System*. The latter is the preferred approach since it
448 provides a single access and maintenance and control point.

449

450 Additionally the *UID* keys may be implemented in physical *XML* syntax in a variety of
451 ways. The architectural needs require that several mechanisms be supported. These
452 mechanisms include, but are not limited to:

453

- 454 • A pure explicit reference mechanism (*XML URN:UID* method),
- 455 • A referential method (*XML URI:UID / namespace:UID*),
- 456 • An object-based reference compatible with W3C Schema (*XML*
457 *URN:complextype name*), and
- 458 • A datatype based reference (for ISO 8601:2000 Date/Time/Number datatyping
459 and then legacy datatyping).

460

461 Examples of each of these in *XML* syntax in the order noted include:

462

- 463 • An *URN:UID* method,
- 464 • An *URI:UID / namespace:UID* method,
- 465 • An *URN:complextype name* method, and
- 466 • An explicit type encoding values as outlined in ISO 8601.

467

468 Additionally, all participating *Components* in ebXML must facilitate multilingual
469 support. Again, a *UID* reference is particularly important here as it provides a language
470 neutral reference mechanism. To enable multilingual support, the ebXML specification
471 must be compliant with Unicode and ISO/IEC 10646 for character set and UTF-8 or
472 UTF-16 for character encoding.

473

474 The underlying ebXML Architecture is distributed in such a manner to minimize the
475 potential for a single point of failure within the ebXML infrastructure. This specifically
476 refers to *Registry and Repository Services* (see *Registry and Repository Functionality*,
477 Section 20 for details of this architecture).

478

479 The implementation of the *FSV* of ebXML, can be categorized as having the following
480 three major phases:

481

482 a) The Implementation Phase

483

484 The implementation phase deals specifically with the procedures for creating an
485 *Application* of the ebXML infrastructure

486

487 b) The Discovery and Deployment Phase

488

489 The Discovery and Deployment Phase covers all aspects of actual discovery of
490 ebXML related resources and self enabled into the ebXML infrastructure.

491

492 c) The Run Time Phase

493

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

496

497 These three phases are now discussed in greater detail.

498

498

499 **13.0 Implementation Phase**

500

501 A *Trading Partner* wishing to engage in an ebXML compliant transaction, must first
 502 request a copy of the ebXML specification. The Specification is then downloaded to the
 503 *Trading Partner*. The *Trading Partner* studies the ebXML specification. The *Trading*
 504 *Partner* subsequently requests to download the *Lexicon* and the *Business Object Library*.
 505 The *Trading Partner* may also request other *Trading Partners' Business Process*
 506 information (stored in its *TPP*) for analysis and review. The *Trading Partner* may also
 507 submit its own *Business Process* information to an ebXML compliant *Registry*.
 508

509 Figure 6 below, illustrates a potential interaction between an ebXML Registry and a
 510 Business Service Interface.

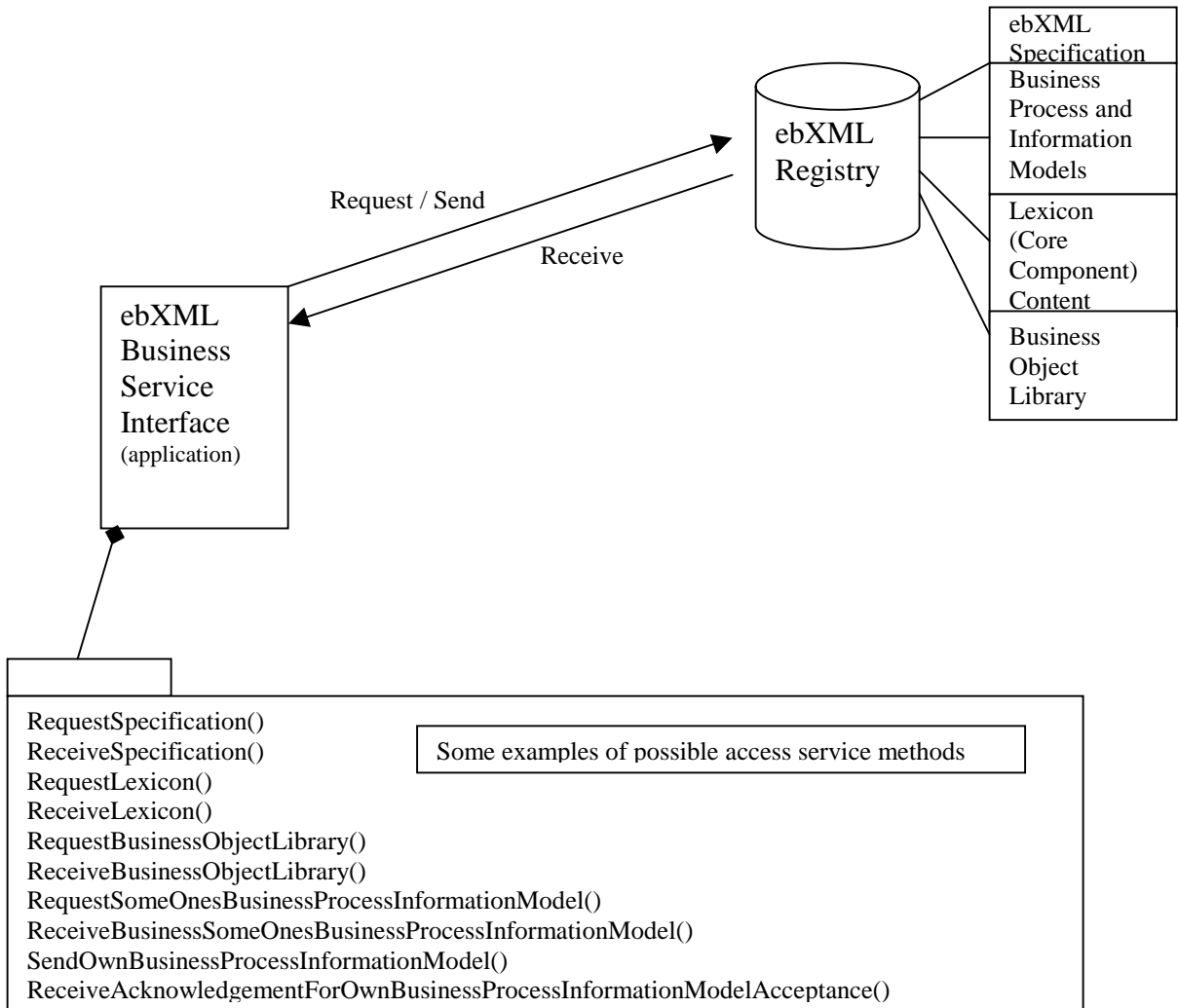


Figure 6: Functional Service View: Implementation Phase

511

512

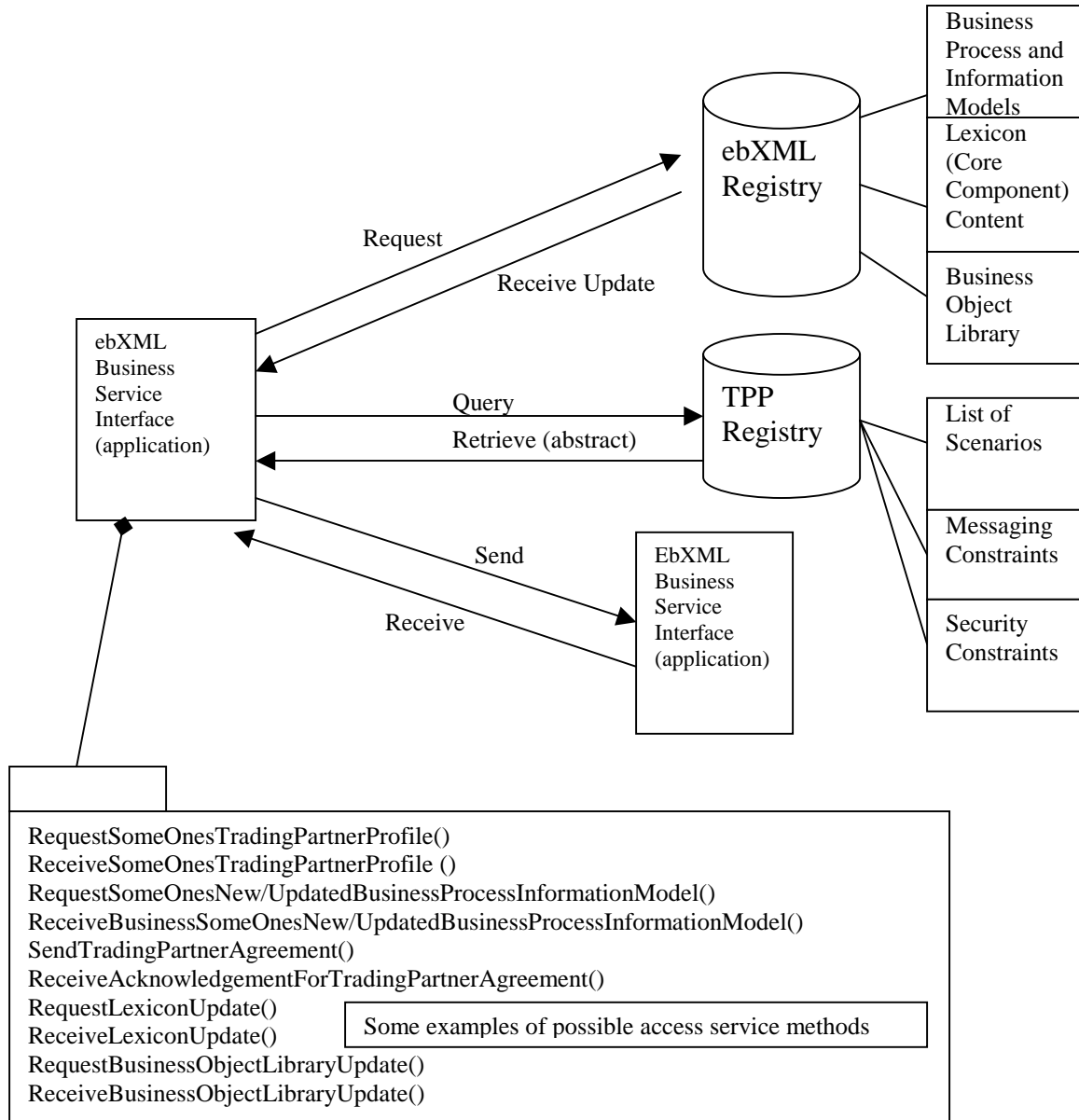
513

514

515 **14.0 Discovery and Deployment Phase**

516

517 A *Trading Partner* who has implemented an ebXML *Business Service Interface* may now
 518 begin the process of discovery and deployment (Figure 7). One possible discovery
 519 method may be to request the *Trading Partner Profile* of another *Trading Partner*.
 520 Requests for updates to *Lexicons*, *Business Object Libraries* and updated or new *Business*
 521 *Process* and information models are also methods which shall be supported by an
 522 ebXML *Application*. This is the phase where *Trading Partners* discover the semantic
 523 meaning of business information being requested by other *Trading Partners*.



524

525

526

Figure 7: Functional Service View: Discovery and Deployment Phase

526 **15.0 Run Time Phase**

527

528 The Run Time phase is the least complex (Figure 8). Note that no *Registry* calls are
 529 required during the Run Time Phase. There are ebXML message instances being sent
 530 and received between *Trading Partners* utilizing the ebXML Messaging Service.

531

532

533

534

535

536

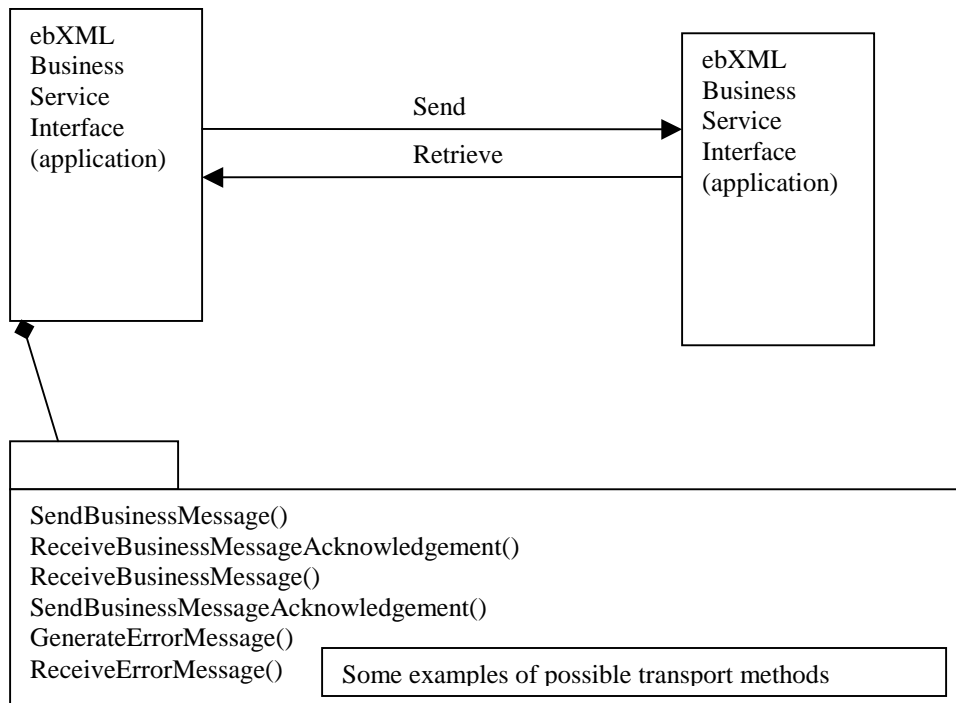
537

538

539

540

541



542

543

544

545

546

547

548

549

550

551

Figure 8: Functional Service View: Run Time Phase

552 **16.0 Trading Partner Information**

553

554 To facilitate the process of conducting *eBusiness*, *SMEs* and other organizations need a
 555 mechanism to publish information about the *Business Processes* they support along with
 556 specific technology implementation details about their capabilities for exchanging
 557 business information. This is accomplished by creating a *Trading Partner Profile (TPP)*.
 558 The TPP is a document which allows a *Trading Partner* to express their minimum
 559 *Business Process* and *Business Service Interface* requirements in a manner where they
 560 can be universally understood by other ebXML compliant *Trading Partners*. The TPP
 561 describes the specific technology capabilities that a *Trading Partner* supports and the
 562 *Service Interface* requirements that need to be met in order to exchange business
 563 documents with that *Trading Partner*. The TPP of the a priori interchange information is
 564 stored in an *ebXML Registry* which provides a discovery mechanism for *Trading*
 565 *Partners* to find one another.

566

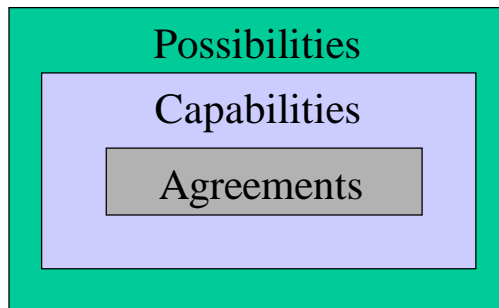
567 **16.1 Support for Trading Partner Agreements**

568 To facilitate the process of conducting electronic business, organizations need a
 569 mechanism to publish information about the *Business Processes* they support, along with
 570 specific technology details about their capabilities for sending and receiving business
 571 documents.

572 ebXML defines the ability for this to be realized under the broad notion of a *Trading*
 573 *Partner Agreement*.

574
 575 A *Trading Partner Agreement (TPA)* is a document that describes: (1) the *Messaging*
 576 *Service* (technology), and (2) the Process (*Application*) requirements that are agreed upon
 577 by two or more parties. A *TPA* is negotiated after the discovery process and is essentially
 578 a snapshot of the specific technology and process related information that two or more
 579 parties agree to use to exchange business information. If any of the parameters of an
 580 accepted *TPA* changes after the agreement has been executed, a new *TPA* shall be
 581 negotiated between all parties.

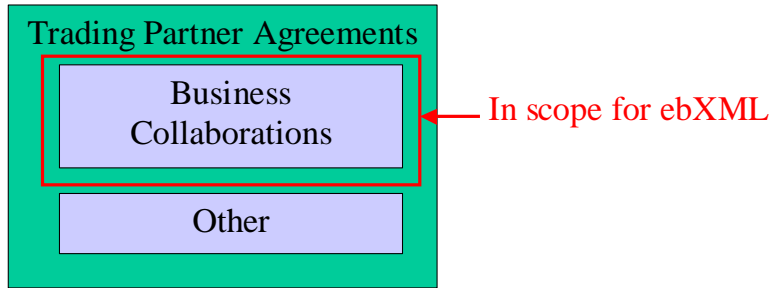
582
 583 Conceptually, ebXML supports a three level view of narrowing subsets to arrive at
 584 agreements for transacting business. The outer-most scope relates to all of the
 585 possibilities that a Partner could do, with a subset of that of what a Partner is capable of
 586 doing, with a subset of what a Partner “will” do.



587
 588
 589 *Figure 9: Three level view of TPA's*
 590

591 ebXML acknowledges the global scope of a *Trading Partner Agreements* to include such
 592 aspects as legal agreement elements and legal ramifications and other trade issues that
 593 are, from an over-arching business perspective, essential elements of “Agreements
 594 between Traders.” ebXML limits its scope within this broad spectrum to addressing the
 595 needs of (electronic) *Business Collaborations*. This provides extensibility for ebXML to
 596 expand to encompass other aspects of *Trading Partner Agreements* on its own or by
 597 embracing other work. Further, the entities engaged in Business Collaborations within
 598 ebXML are referred to as *Partners*. Business Collaborations are the first order of support
 599 that can be claimed by ebXML Partners. This “claiming of support” for specific Business
 600 Collaborations is facilitated by a distinct profile defined specifically for publishing, or
 601 advertising in a directory service, like the ebXML *Registry/Repository* or other available
 602 similar services.

603



604
605
606
607

Figure 10: Scope for TPA's

608 **17.0 Business Process and Information Modeling**

609
610 **17.1 Overview**

611 The purpose of the *Business Process and Information Modeling* specification is to enable
612 the modeling of the business relationships between partners in a shared *Business Process*,
613 and their interaction and information exchange as they each perform roles within that
614 process. In general terms, a *Business Process* is defined as a sequenced set of business
615 transactions. A business transaction is a clearly defined exchange of *Business Messages*
616 resulting in a new legal or commercial state between the two partners. The business
617 *Semantics* of each commercial transaction are defined in terms of the *Business Objects*
618 affected, and the commitment(s) formed or agreed. The technical *Semantics* of each
619 commercial transaction are defined in terms of a 'handshake' protocol of required
620 message (signal) exchanges.

621
622 **17.2 Position within overall ebXML Architecture**

623 The *Business Process and Information Modeling* specification has important semantic
624 relationships to the *Core Component* specification and to the *Trading Partner*
625 *Specification*. In addition, the business models produced are registered within an ebXML
626 *Registry/Repository*.

627
628

628

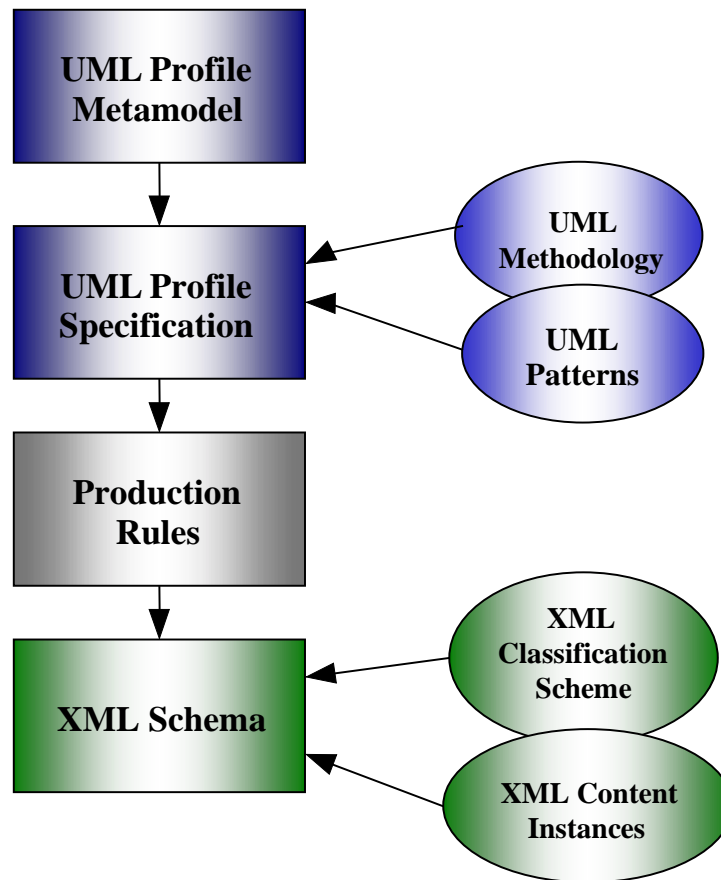
629 **17.3 Business Process and Information Modeling Functionality**

630 The *Business Process and Information Modeling* specification supports *UML Business*
 631 *Process* and information modeling, along with conversion of *UML* models into *XML*, and
 632 direct access to the *XML* expression of the model. Within the *UML* modeling system the
 633 ebXML specification provides a *UML* profile, a set of recommended diagrams, and a
 634 selected methodology to follow in constructing those diagrams. For the conversion of
 635 *UML* to *XML* the specification provides a set of production rules. For further
 636 standardization, the specification provides a set of core processes, and a set of patterns
 637 from which to compose new process definitions.

638

639 The ebXML *Metamodel* specifications constitute a set of *XML* structures that can be
 640 populated and stored in an *ebXML Registry and Repository System*. The *XML* structures
 641 may utilize a classification system with *UID* reference linkages which are compatible
 642 with the *Registry and Repository* architecture requirements.

643



644

645

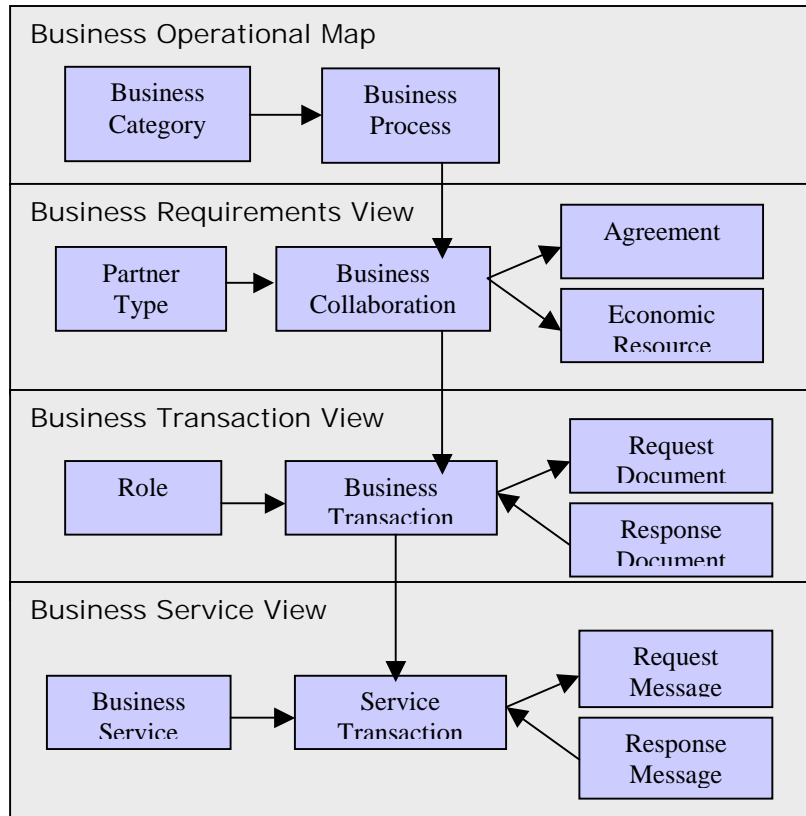
646 *Figure 11: Relationship of UML to ebXML Metamodel content representation.*

647

648 **17.4 The Business Process and Information Metamodel**

649 The *Business Process and Information Metamodel* is composed of four layers (see Figure
 650 12 below). The top layer consists of the Business Operational Map which supports the

651 process of relating different *Business Processes* to each other into a map as well as the
 652 categorization of *Business Processes* by business or process area. The next layer, the
 653 Business Requirements View, supports definition of the partner type which partakes in a
 654 step within a *Business Process* along with the business agreements resulting from or
 655 governing that step and the economic resource commitment or exchange resulting from
 656 that step. The Business Transaction View supports the specification of business
 657 transactions in terms of exchanged business documents. The bottom layer, the Business
 658 Service View, captures the syntax and *Semantics* of *Business Messages* and their
 659 exchange between business services.



683 **Figure 12: Business Process and Information Metamodel Architecture**

686 **17.5 Interfaces and Relationship**

687 The interface from a *Business Process and Information Metamodel* to other parts of the
 688 ebXML Architecture, or to other tools and environments outside the scope of the ebXML
 689 specifications, is an *XML* document representing the *Business Process and Information*
 690 *Metamodel*. Specifically, the interface between the *Business Process and Information*
 691 *Metamodel* and the *Trading Partner* model is the part of such an *XML* document that
 692 represents the business transactional layer of the *Business Process Metamodel*. The
 693 expression of the sequence of commercial transactions in *XML* is shared between the
 694 *Business Process* and *Trading Partner* models.
 695

696 **17.6 Relationship to Trading Partner Agreements**

697 The interface between the *Business Process and Information Metamodel* and the *Trading*
698 *Partner* specification is the sequence of business transactions, the commercial business
699 itself, and the message exchange in support of the business transaction. The profile of a
700 *Trading Partner* defines that partner's functional and technical capability to support one
701 or more roles in a *Business Process*. The agreement between two *Trading Partners*
702 defines the actual conditions under which the two partners will conduct business
703 transactions together.

704

705 **17.7 Relationship to Core Components**

706 A *Business Process* can be seen as a series of actions on entities within an enterprise,
707 interleaved with a set of communications with parties outside the enterprise. The
708 communication between the parties is the shared part of the *Business Process*. This is the
709 focus of ebXML.

710

711 The entities within an enterprise are called *Business Entities*. Their data structure may be
712 represented using *Business Objects*.

713

714 The communication with parties outside the enterprise takes place through an exchange
715 of business documents.

716

717 Both *Business Objects* and business documents are composed from *Core Components*, re-
718 useable low-level data structures.

719

720 The exact composition of a *Business Object* or a business document is guided by a set of
721 contexts derived from (among other sources) the *Business Process*.

722

723 **18.0 Core Component Functionality**

724

725 A *Core Component* captures information about a real world (business) concept, and
726 relationships between that concept and other business concepts.

727

728 A *Core Component* can be either an individual piece of business information, or a natural
729 "go-together" family of business information pieces. It is 'Core' because it occurs in
730 many different areas of industry/business information interaction.

731

732 A *Core Component* may contain:

733

- Another *Core Component* in combination with one or more individual business information pieces.

734

- Other *Core Components* in combination with zero or more individual business information pieces.

735

736

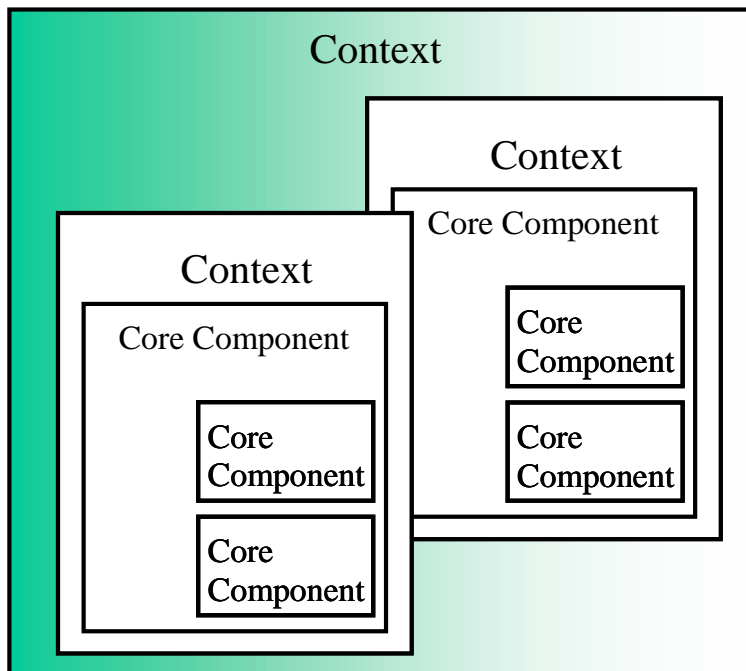
737

738 A *Core Component* needs to contain either attribute(s) or be part of another *Core*
 739 *Component*, thus specifying the precise context or combination of contexts in which it is
 740 used.

741
 742 Context may be structural, identifying the placement of a *Core Component* within
 743 another *Core Component*. It may be a combination of structural contexts when the *Core*
 744 *Component* is re-used at different layers within another *Core Component*.

745
 746 Context will also be defined by the *Business Process* model, which defines the instances
 747 in which the *Business Object* occurs.

748
 749



750
 751
 752
 753

Figure 13: Core components as contextual items.

754 The pieces of information, or *Core Components*, within a generic *Core Component* may
 755 be either mandatory, or optional. A *Core Component* in a specific context or combination
 756 of contexts may alter the fundamental mandatory/optional cardinality.

757
 758 Individual *Core Components* will in general match the “data list” part of *Business*
 759 *Objects*.

760 **19.0 Business Object Functionality**

761 **19.1 Overview**

762 The term *Business Object* is used in two distinct ways in ebXML, with different
 763 meanings for each usage:
 764

765

766

- In a business model, *Business Objects* describe a business itself, and its business context. The *Business Objects* capture business concepts and expresses an abstract view of the business's "real world" functions.

767

768

769

770

- In a business software *Application* or service, *Business Objects* reflects how business concepts are represented in software. The abstraction here reflects the transformation of business ideas (processes) into a software implementation.

771

772

773

774

Within the context of ebXML, only *Business Objects* represented in *Business Processes* and information models are of relevance.

775

776

777

19.2 Business Objects in Business Process and Information Models

778

A *Business Object* describes a thing, concept, process or event in operation, management, planning or accounting of a business or other organization. It is a conceptual *Object* that has been specified for the purpose of directly describing and representing, and thus serving, a business concept or purpose. The focus/subject is the business subject/concept being modeled.

779

780

781

782

783

784

A *Business Object* in this usage is a specification for a kind of *Object* which may exist in one or more business domains. The specification of a *Business Object* may include attributes, relationships, and actions/events that apply to these *Objects*. These *Business Object Models* may exist regardless of the existence of information systems, *Applications*, software design or program code. They are independent of information systems because *Business Object Models* directly reflect and abstract "real world" business concepts and scenarios. Thus *Business Object Models* are defined independently of *Application* systems.

785

786

787

788

789

790

791

792

793

The primary concern when creating *Business Object Models* is capturing common business *Semantics* and having a common idea or concept that is usable by different parts of a business or by different independent businesses.

794

795

796

797

19.3 Common Business Objects

798

A *Common Business Object (CBO)* is a *Business Object* that is specified in more than one Domain. For the purposes of defining *CBOs*, a domain is defined as an industry sector.

799

800

As with all *Business Objects* in general, the most important issue with *CBOs* is a common concept and mutually agreed upon structure.

801

802

802 **20.0 Registry and Repository Functionality**

803

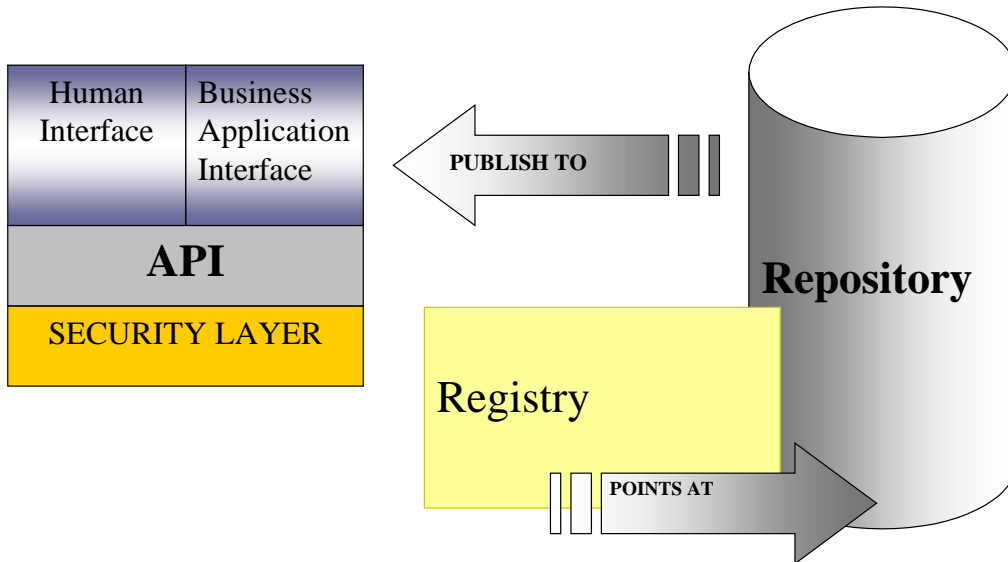
804 **20.1 Overview**

805 An *ebXML Registry* provides a set of distributed services that enable the sharing of
806 information between interested parties for the purpose of enabling *Business Process*
807 integration between such parties by utilizing the ebXML specifications. The shared
808 information is maintained as objects in an *ebXML Repository* which is managed by
809 *ebXML Registry Services*. Access to an *ebXML Repository* is provided by the interfaces
810 (APIs) exposed by *Registry Services*.

811

812 Therefore, architecturally the *Registry* and *Repository* are tightly coupled *Components*.
813 The *Registry* provides the access services interfacing, the information model and
814 reference system implementation, while a *Repository* provides the physical backend
815 information store. For example, an *ebXML Registry* may provide a *Trading Partner*
816 *Profile* from the *Repository* in response to a query; or an *ebXML Repository* may contain
817 reference DTD's or Schemas that are retrieved by the *Registry* as a result of searching a
818 metadata classification of the DTD's or Schemas. Figure 14 provides an overview of this
819 configuration.

820



821

822

Figure 14: Registry / Repository interaction overview.

823

824

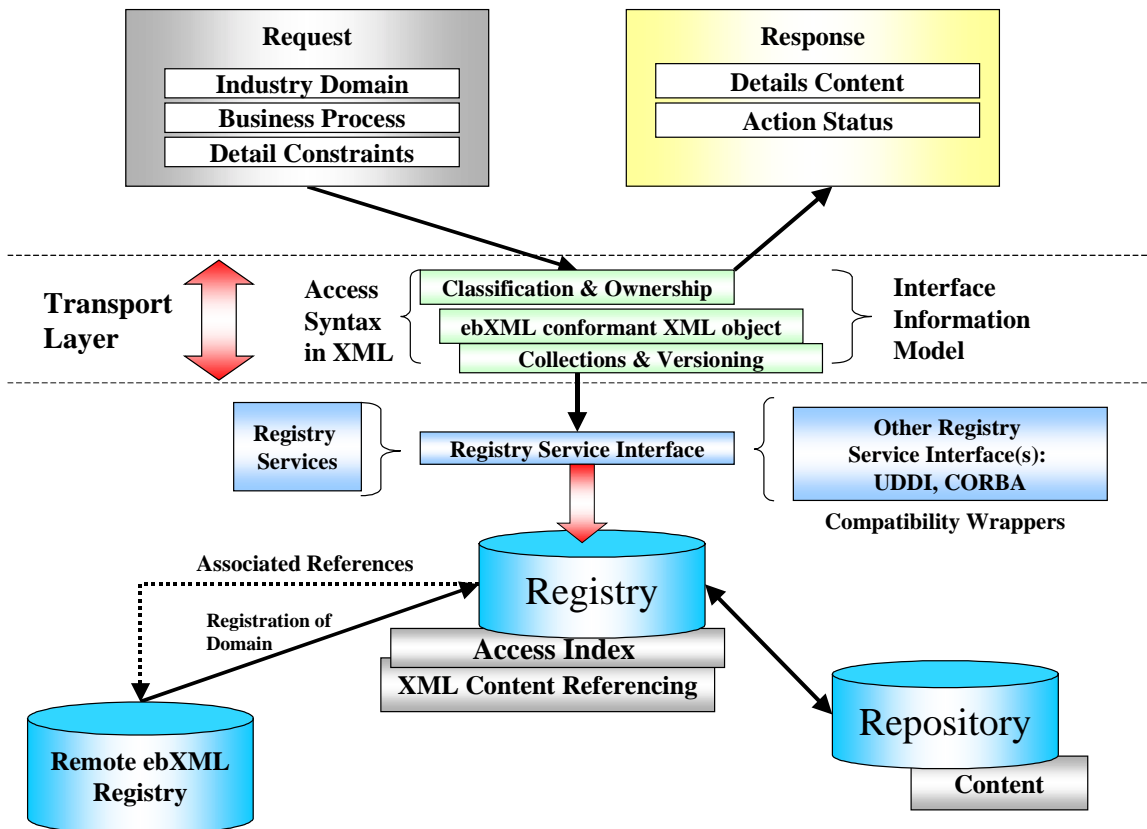
825

826

826 **20.2 Information Model & Interface Constrains**

827 In order to accurately and consistently store and retrieve information a *Registry* requires a
 828 formal architecture that includes an information model. Traditional relational SQL
 829 databases have a very simple information model that includes tables, indexes and column
 830 definitions for content to be stored into them. However, when using *XML* structures to
 831 store and manage content we potentially have an infinite variety of ways to present
 832 information to a *Registry*. We therefore to ensure there are particular aspects of those
 833 structures that allow us to manage them within the *Registry*, and that also these aspects
 834 are linked to the access methods that will be used to interface to the *Registry*. Providing
 835 the mechanisms to support the business functional capabilities expressed in these *XML*
 836 structures and content and ensuring it functions correctly is the role of the information
 837 model.

838
 839 The information model for the *ebXML Registry* is an extension of the existing OASIS
 840 *Registry* information model, specifically tailored for the storage and retrieval of business
 841 information content, whereas the OASIS model is a superset designed for handling
 842 extended and generic information content. As such the *ebXML Registry* information
 843 model is designed to make it easier to implement and to provide explicit ebXML
 844 *Metamodel* compliant instance structures to facilitate accessing and storing ebXML
 845 content.
 846



847
 848
 849

Figure 15: Registry / Repository Architecture.

850

851 A *Registry* maintains the metadata for a registered object, and a *Repository* maintains the
852 file containing a registered object. The *Registry* and *Repository* are tied together in that
853 the metadata for a registered object in the *Registry* includes a globally unique locator for
854 a file, in some *Repository*, that contains the registered object.

855

856 A *Registry* Item contains information that identifies, names, and describes each registered
857 object, gives its administrative and access status, defines its persistence and mutability,
858 classifies it according to pre-defined classification categories, declares its file
859 representation type, and identifies the submitting and responsible organizations.

860

861 Related to this the existing ISO11179/3 work on business semantic content *Registry*
862 implementations is used to provide a model for the *ebXML Registry* implementation.
863 Again the approach is to take a tailored subset of the ISO11179 functionality that is
864 applicable to the *ebXML Registry* requirements, and to make implementing *ebXML*
865 *Registry* systems simpler for vendors to implement and librarians to manage.

866

867 Combined together these reference specifications are then exposed via the *Registry*
868 Interface system itself. The *Registry* interface is the architectural component that
869 provides both a machine-to-registry automated access system, and also a human-to-
870 *Registry* interactive visual access system. The *Registry* interface system is designed to
871 be a primitive *XML*-based interface that is transport layer neutral. However, the
872 reference implementation of the *Registry* interface is built using the ebXML Transport
873 layer facilities only. Similarly the query syntax used by the *Registry* access mechanisms
874 is designed to be a neutral syntax based solely in *XML* syntax, and independent of the
875 physical product implementation of the backend *Repository System*.

876

877 **20.3 Formal Functional Overview**

878 A *Registry/Repository System* may have many deployment models that yield the same
879 functionality. The initial specification and implementation will define the minimal
880 functional requirements that a *Registry/Repository System* shall provide to facilitate its
881 role in the ebXML infrastructure. It is expected that future specifications and
882 implementations will evolve into more complex systems.

883

884 All interaction between a *Registry* clients and the *Registry* are treated as business
885 transactions between parties. Thus the processes supported by the *Registry* are described
886 in terms of:

887

- 888 • A special *TPA* between the *Registry* and *Registry* clients.
- 889 • A set of business functional processes involving the *Registry* and *Registry* clients.
- 890 • A set of *Business Messages* exchanged between a *Registry* client and the *Registry*
891 as part of a specific business functional process.
- 892 • A set of primitive interface mechanisms to support the *Business Messages* and
893 associated query and response mechanisms.
- 894 • A special *TPA* for between one *Registry* interoperating with another *Registry*.

- 895 • A set of functional processes involving *Registry* to *Registry* interactions.
- 896 • A set of error responses and conditions with remedial actions.

897

898 The *Registry* interactions supported here are intended to be a limited subset of the full
899 requirements as defined by the ebXML Requirements documents. The architecture
900 described here is based on supporting the conceptual ebXML architecture and business
901 interactions as defined in Section 8 of this specification. Some of the extended
902 functionality deferred to a subsequent phase includes transformation services, workflow
903 services, quality assurance services and extended security mechanisms.

904

905 **20.4 Sample Objects Residing in a Repository and Managed by a Registry**

- 906 • **Schema:** These objects are documents that represent the schema (*XML* DTD, etc.)
907 for *XML* documents.

908

- 909 • **Process:** These are objects that represent a *Business Process*. These could include
910 a process description in an *XML* form such as *XMI* or could be actual software
911 *Components* (e.g. Java Classes) that could represent an implementation of a
912 *Business Process*.

913

- 914 • **Trading Partner Profile:** These are *XML* documents that provide information
915 about a *Party* interested in participating in B2B interaction.

916

- 917 • **Reference Content:** there are two types of reference content, those that describe
918 the reference information model and classification systems within the *Registry*
919 itself (schemas), and those that categorize industry business information (*XML*
920 document instances). The later are often standard information sets that can be
921 expected to reside in and be supported by the *Registry* information model, such as
922 ISO reference datatypes, ISO reference code tables and similar open public
923 definitions.

924

- 925 • **Any object with metadata:** Elements provide standard metadata about the object
926 being managed in the *Repository*. Note that the object metadata is separate from
927 the object itself, thus allowing the *ebXML Registry* to catalog arbitrary objects.

928

929 **20.5 Registry Management of Repository Objects and Metadata**

930 *Registry* messages shall exist to create, modify and delete *Repository Objects* and their
931 metadata. Appropriate security *Protocols* shall be deployed to offer authentication and
932 protection for the *Repository* when accessed by the *Registry*.

933

934 Additionally all content stored into a *Registry/Repository* is implicitly public and open
935 information. Therefore parties submitting information to an *ebXML Registry* should
936 ensure that they have appropriate intellectual rights and permissions to submit this
937 information. An *ebXML Registry* will provide administrative access rights to ensure only
938 the submitting organization has formal access to change the content, however all other
939 retrieval rights will be open. For this reason, *TPAs*, which are necessarily proprietary to

940 *Trading Partners* will not be stored within an ebXML *Registry*, only the public *TPP*
 941 details will be stored within an ebXML *Registry*.

942

943 **20.6 Querying Registries and Returning Repository Objects and Metadata**

944 A *Registry* query mechanism shall be employed to query for *Repository Objects* and their
 945 metadata by either an *Application* automated interface or a Human software GUI
 946 interface.

947

948 *Repository Objects* and their metadata shall be made available by ebXML messages sent
 949 to the *Registry* (typically an *Application* requestor service).

950

951 *Repository Objects* and their metadata can also be addressable where applicable as an
 952 XML based URI reference using only HTTP for simple direct access.

953 Each *Repository Object* is identified by a Unique Identifier key (see Section 12 for an
 954 introduction on *UID* key mechanisms). A query on a *Unique Identifier (UID)* returns one
 955 and only one *Repository* object.

956

957 Metadata queries perform an object search based on the metadata defined for (but
 958 maintained outside) a managed object.

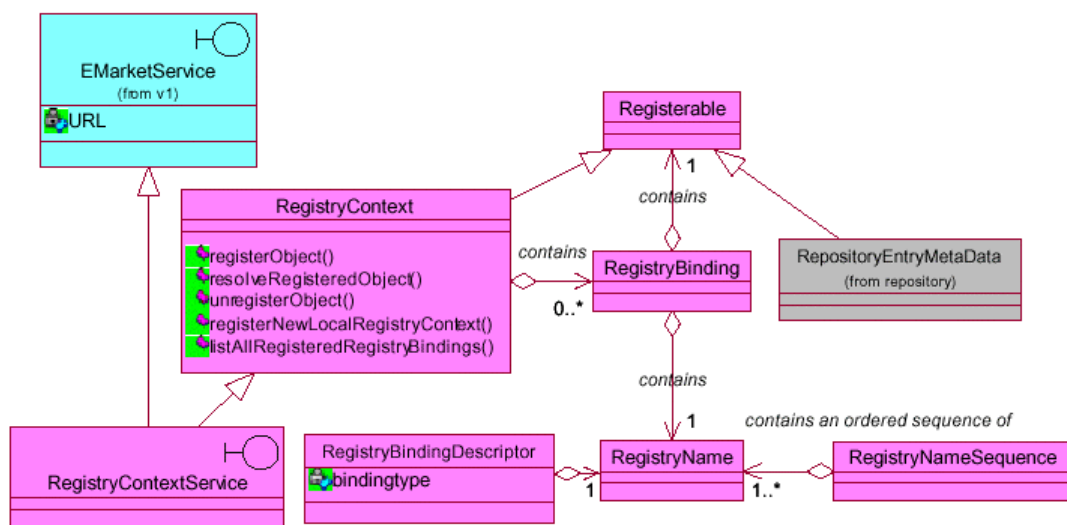
959

960 Browse and drill down queries are expected to be the primary *Use Case* for querying the
 961 *Registry* by Web based human interactions. In this scenario, a user browses the
 962 *Repository* content using a Web browser via a HTTP protocol. The user may initially
 963 browse and traverse the content based on the built-in classification schemes.

964

965 **20.7 Registry to Registry Interfacing Model**

966 Since ebXML Registries are distributed each *Registry* may potentially interact with and
 967 cross-reference to another ebXML *Registry*. The following diagram provides an example
 968 of the architectural *Components* that facilitate these mechanisms.



969
 970

Figure 16: Registry to Registry Service Components.

971

972 Referencing Figure 16 above, the following apply:

973

974 A RegistryName to RegistryObject association is called a RegistryBinding. A

975 RegistryBinding is always defined relative to a RegistryContext. A RegistryContext is an

976 object that contains a set of RegistryBindings in which each RegistryName is unique.

977 Different RegistryNames can be bound to a RegisteredObject in the same or different

978 RegistryContexts at the same time.

979

980 A RegistryObject specialization (a RegistryContext or a RepositoryEntryMetaData),

981 which is bound into a RegistryContext is unaware of the fact that it has been associated to

982 a RegistryName via a RegistryBinding, and that the RegistryBinding may be bound into a

983 RegistryContext (not navigable).

984

985 A RegistryName is used to identify the binding within the RegistryContext for which it

986 may be bound. A RegistryNameSequence is an ordered set of RegistryNames that can be

987 used to resolve a RegisteredObject from a given target RegistryContext.

988 RegistryContextService is RegistryContext boundary interface and is an EMarketService.

989 For the extent of the model scope of this document, a URL is inherited and is used to

990 facilitate distribution of RegistryContexts through URL addressing.

991

992 A RegistryBindingDescriptor describes a RegistryBinding by identifying the type of

993 binding and the RegistryName. RegistryBindingDescriptors are returned on list messages

994 on RegistryContexts.

995

996 The architecture of the ebXML metadata classification system within the *ebXML Registry*

997 itself will be extended (see Figure 16 above). These extensions will support references to

998 domains that are not directly managed by that *Registry*, and its associated *Repository*

999 store.

1000

1001

1001 **20.8 Registry/Repository Business Scenario Example**

1002 In addition to the use of the *ebXML Registry* as a means to facilitate and enable the core
 1003 architecture of ebXML compliant information exchanges, a *Registry/Repository* may also
 1004 be used to facilitate business functional implementations. An example would be a
 1005 network of *Trading Partners* similar to a telephone directory Yellow Pages system where
 1006 businesses can be categorized by services that they provide.
 1007

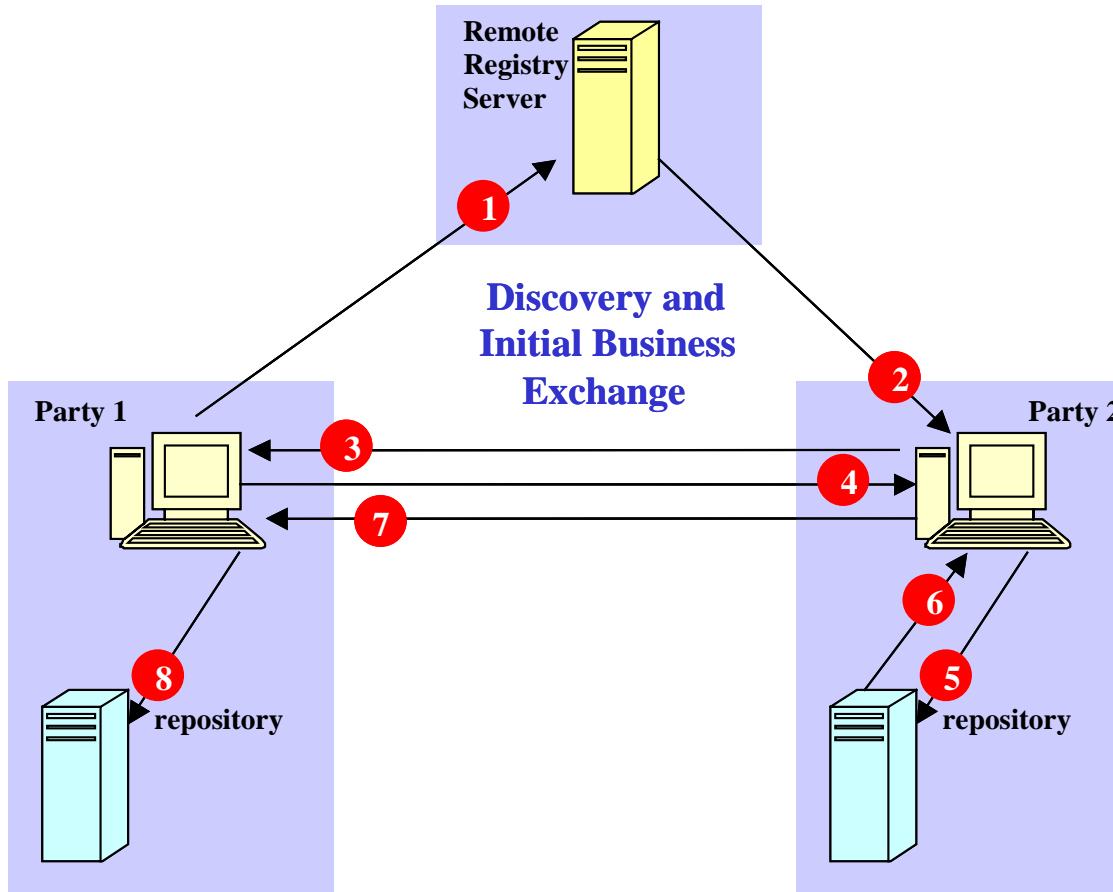


Figure 17: Trading Partner Discovery.

1008
 1009
 1010
 1011

1012 **21.0 Messaging Service Functionality**

1013
 1014

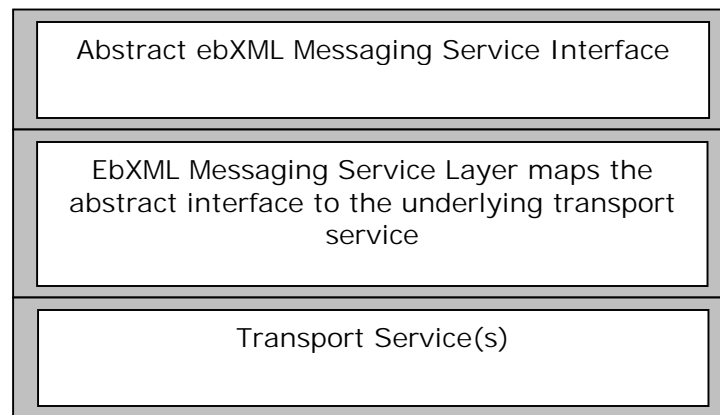
1014 **21.1 Overview**

1015 The *ebXML Messaging Service* provides for the secure, reliable exchange of *ebXML*
 1016 *Messages* between *Parties* over various transport *Protocols* (SMTP, HTTP/S, FTP, etc.).
 1017 The *Messaging Service* specification defines the MIME packaging and ebXML message
 1018 *Header* information required by the *ebXML Messaging Service* to enable interoperable
 1019 exchange of ebXML compliant messages.
 1020

1021 The *ebXML Messaging Service* supports all messaging between distributed *Components*
 1022 of the ebXML system including *Registry/Repository* and ebXML compliant *Applications*.

1023 It utilizes and enforces the "rules of engagement" defined in a *Trading Partner*
 1024 *Agreement (TPA)*. The *ebXML Messaging Service* supports simplex (one-way) and
 1025 request/response (either synchronous or asynchronous) message exchange and can be
 1026 mapped onto any transport service capable of transporting MIME (further discussed in
 1027 section).

1028
 1029 The *ebXML Messaging Service* is conceptually broken down into three parts: (1) an
 1030 abstract *Service Interface*, (2) functions provided by the *Messaging Service Layer*, and
 1031 (3) the mapping to underlying transport service(s). The relation of the abstract interface,
 1032 *Messaging Service Layer*, and transport service(s) are shown in Figure 18 below:



1033
 1034
 1035
 1036
 1037
 1038
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
Figure 18: ebXML Messaging Service

1048 **21.2 Abstract ebXML Messaging Service Interface**

1049 The ebXML Message Service provides ebXML with an abstract interface whose
 1050 functions, at an abstract level, include:

- 1051
- 1052 • Send – send an *ebXML Message* – values for the parameters are derived from the
 - 1053 *ebXML Message Headers*.
 - 1054 • Receive – indicates willingness to receive an *ebXML Message*.
 - 1055 • Notify – provides notification of expected and unexpected events.
 - 1056 • Inquire – provides a method of querying the status of the particular ebXML
 - 1057 Message interchange.

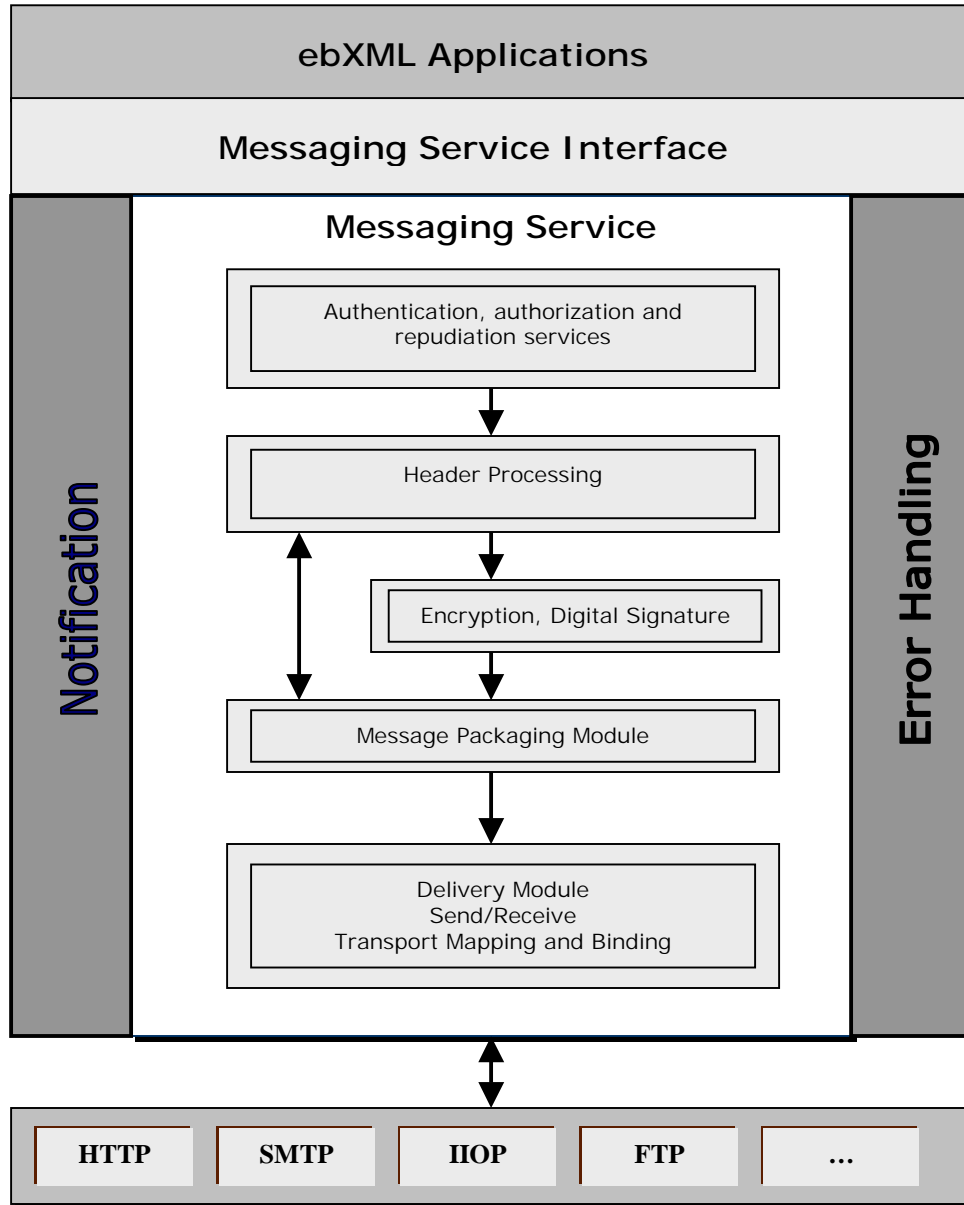
1058
 1059 **21.3 ebXML Messaging Service Layer Functions**

1060 The *ebXML Messaging Service Layer* provides all of the services and functionality
 1061 needed to manage the entire lifecycle of *ebXML Messages*. Functions provided by this
 1062 layer include:

- 1063
- 1064 • The ability to construct and validate proper *ebXML Messages*.
 - 1065 • Enforcing the "rules of engagement" as defined by two parties in a *Trading*
 - 1066 *Partner Agreement* (including security and *Business Process* functions related to
 - 1067 message delivery). The *Trading Partner Agreement* defines the acceptable
 - 1068 behavior by which each *Party* agrees to abide. The definition of these ground

- 1069 rules can take many forms including formal *Trading Partner Agreements*,
1070 interactive agreements established at the time a business transaction occurs (e.g.
1071 buying a book online), or other forms of agreement. There are *Messaging Service*
1072 *Layer* functions that enforce these ground rules. Any violation of the ground rules
1073 result in an error condition, which is reported using the appropriate means.
- 1074 • Support for the following reliability options:
 - 1075 ○ "Best Effort" delivery
 - 1076 ○ "Once and only once" delivery
 - 1077 ○ Synchronous or Asynchronous messaging
 - 1078 ○ Request/Response processing
 - 1079 ○ Fire 'n forget processing
 - 1080 ○ Allow for "multipart" message delivery
 - 1081 • Perform all security related functions including:
 - 1082 ○ Identification
 - 1083 ○ Authentication (verification of identity)
 - 1084 ○ Authorization (access controls)
 - 1085 ○ Privacy (encryption)
 - 1086 ○ Integrity (message signing)
 - 1087 ○ Non-repudiation
 - 1088 ○ Logging
 - 1089 • Interface with internal systems including:
 - 1090 ○ Routing of received messages to internal systems
 - 1091 ○ Error notification
 - 1092 • Administrative services including:
 - 1093 ○ Notification, both system-to-system and system-to-human (via pagers or
1094 e-mail)
 - 1095 ○ Track and report the status of message exchanges for auditing and
1096 diagnostic purposes
 - 1097 ○ Logging of service related errors
 - 1098 ○ Access to *Partner Agreement* information
 - 1099 ○ Status inquiry
 - 1100 • Transport bindings:
 - 1101 ○ Functions to enable the delivery of messages over various transport
1102 services (e.g. SMTP, FTP, HTTP, etc.)
 - 1103
 - 1104

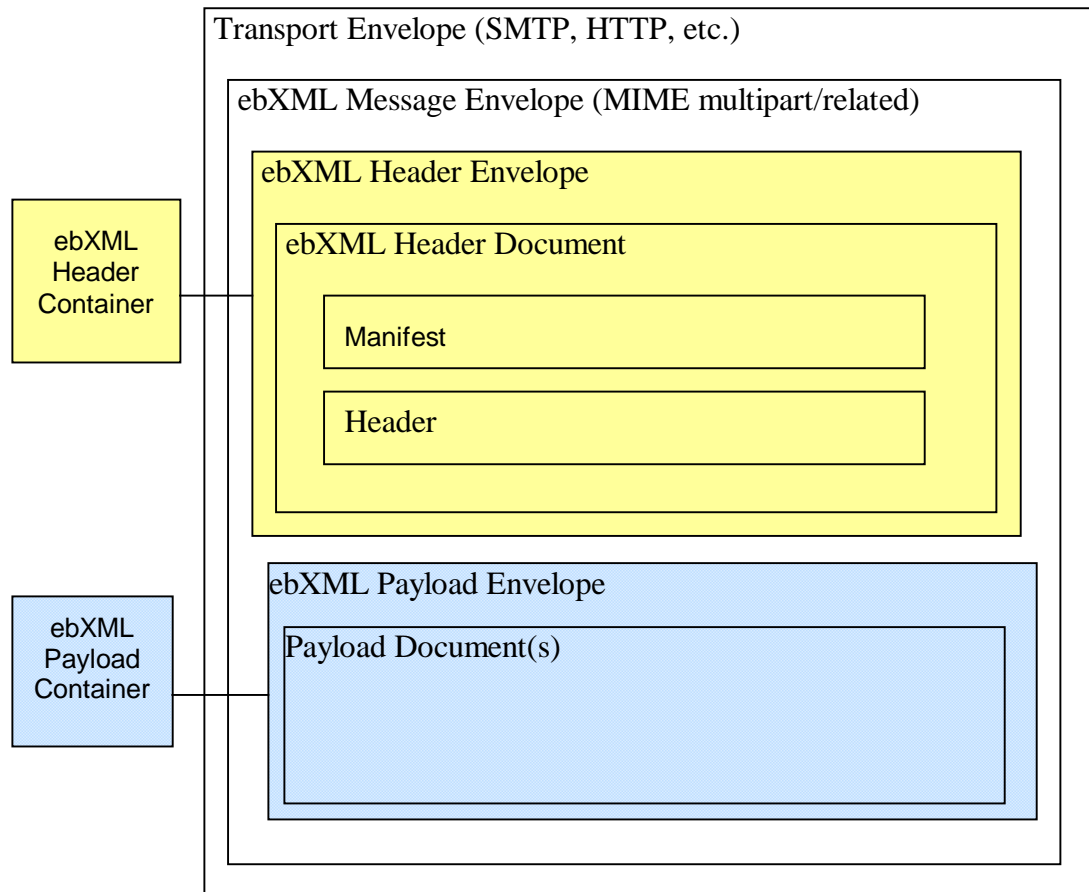
1104 The following diagram depicts a logical arrangement of the functional modules that exist
1105 within the ebXML *Messaging Service* architecture. These modules are arranged in a
1106 manner to indicate their inter-relationships and dependencies. This architecture diagram
1107 illustrates the flexibility of the ebXML Messaging Service, reflecting the broad spectrum
1108 of services and functionality that may be implemented in an ebXML system.
1109
1110



1141
1142
1143
1144
1145
1146
1147
1148 **Figure 19: The Messaging Service Architecture**
1149

1149 **21.4 ebXML Message Structure and Packaging**

1150 Figure 20 below illustrates the logical structure of an ebXML compliant message.



1151 *Figure 20: ebXML Message Structure*

1152

1153

1154

1155 An *ebXML Message* consists of an optional transport *Protocol* specific outer
 1156 *Communication Protocol Envelope* and a *Protocol* independent *ebXML Message*
 1157 *Envelope*. The *ebXML Message Envelope* is packaged using the MIME multipart/related
 1158 content type. MIME is used as a packaging solution because of the diverse nature of
 1159 information exchanged between *Partners* in *eBusiness* environments. For example, a
 1160 complex B2B business transaction between two or more *Trading Partners* might require
 1161 a payload that contains an array of business documents (*XML* or other document
 1162 formats), binary images, or other related *Business Objects*.

1163

1164 The *ebXML Message Envelope* is used to encapsulate the *Components* of an ebXML
 1165 compliant message. This structure effectively separates ebXML header information from
 1166 the payload content of the message. The separation of *Header* and *Payload* containers
 1167 promotes system efficiency, as the *ebXML Messaging Service* only needs to access
 1168 *Header* information to process the message. This provides a flexible mechanism for
 transparently passing diverse *Payloads* to appropriate business services without having to

1169 process them within the *Messaging Service* framework. It also allows encrypted and/or
1170 signed *Payloads* to be exchanged and forwarded with no processing overhead.

1171

1172 The *ebXML Payload Container* is an optional part of an *ebXML Message*. If a *Payload* is
1173 present in an *ebXML Message*, the *ebXML Payload Envelope* serves as the container for
1174 the actual content (*Payload*) of the *ebXML Message*. The *ebXML Payload Envelope*
1175 consists of a MIME header portion and a content portion (the *Payload* itself). The
1176 *ebXML Messaging Service* does not limit in any way the structure or content of payloads.

1177

1178 **22.0 Conformance**

1179

1180 **22.1 Overview**

1181 The objectives of this section are to:

- 1182 a) Ensure a common understanding of conformance and what is required to claim
1183 conformance;
- 1184 b) Promote interoperability and open interchange of *Business Processes* and
1185 messages;
- 1186 c) Promote uniformity in the development of conformance tests.

1187

1188
1189
1190 ebXML conformance is defined in terms of conformance to ebXML, conformance to
1191 each of the component specifications for ebXML, and conformance to this (Technical
1192 Architecture) specification.

1193

1194 All ebXML specifications shall contain a conformance clause. The conformance clause
1195 specifies explicitly all the requirements that have to be satisfied to claim conformance to
1196 that specification. These requirements may be applied and grouped at varying levels
1197 within each specification.

1198

1199 **22.2 Conformance Requirements**

1200 Types of conformance requirements can be classified as:

1201

- 1202 a) Mandatory requirements: these are to be observed in all cases;
- 1203 b) Conditional requirements: these are to be observed if certain conditions set out in
1204 the specification apply;
- 1205 c) Optional requirements: these can be selected to suit the implementation, provided
1206 that any requirement applicable to the option is observed.

1207

1208 Furthermore, conformance requirements in a specification can be stated:

1209

- 1210 • Positively: they state what shall be done;

1211

- 1213 • Negatively (prohibitions): they state what shall not be done.

1214

1215 **22.3 General Framework of Conformance Testing**

1216 The objective of conformance testing is to establish a set of criteria that enable vendors to
1217 implement compatible and interoperable systems built on the ebXML foundations.

1218 Since ebXML consists of many facets and *Components*, ebXML conformance shall take
1219 into account different layers and levels. These levels will be hierarchical and recursive, so
1220 conformance to a higher level will include conformance to a lower level.

1221

1222 Implementations and *Applications* shall be tested to verify their conformance to ebXML
1223 Specifications.

1224

1225 Publicly available test suites from vendor neutral organizations such as OASIS and NIST
1226 should be used to verify the conformance of ebXML implementations, *Applications*, and
1227 *Components* claiming conformance to ebXML. This will ensure that they are compliant
1228 with the base ebXML criteria. Live benchmark implementations may be available to
1229 allow vendors to test their products for interface compatibility and conformance.

1230

1231 Additional items of note include:

1232

1233 a) Extended implementations may include support for more than just the base
1234 ebXML *Protocols*, including other popular or emerging formats, such as legacy
1235 *EDI* or *Messaging Services* such as Java Messaging Service (JMS)
1236 implementations.

1237

1238 b) Each ebXML working group will be responsible to coordinate with and determine
1239 what it means to conform to their specification and what should be included in the
1240 appropriate Conformance test suite(s).

1241

1241 **Appendix A: Example ebXML Business Scenarios**

1242

1243 **Definition**

1244 This set of Scenarios defines how ebXML compliant software could be used to
1245 implement popular, well-known *eBusiness* models.

1246 **Scope**

1247 These Scenarios are oriented to properly position ebXML specifications as a convenient
1248 mean for Companies to properly run *eBusiness* over the Internet using open standards.
1249 They bridge the specifications to real life uses.

1250 **Audience**

1251 Companies planning to use ebXML compliant software will benefit from these Scenarios
1252 because they will show how these companies may be able to implement popular business
1253 scenarios onto the ebXML specifications.

1254 **List**

- 1255 e) Two Partners set-up an agreement and run the associated electronic exchange.
- 1256 f) Three or more partners set-up a *Business Process* implementing a supply-chain
1257 and run the associated exchanges
- 1258 g) A Company sets up a Portal which defines a *Business Process* involving the use
1259 of external business services.
- 1260 h) Three or more parties engage in multi-Party *Business Process* and run the
1261 associated exchanges.

1262

1263 **Scenario 1: Two Partners set-up an agreement and run the associated** 1264 **exchange**

1265 In this scenario:

- 1266 • Each partner defines its own *Party Profile*.
1267 Each *Party Profile* references:
 - 1268 • One or more existing *Business Process* found in the ebXML *Repository*
 - 1269 • One of more Message Definitions. Each Message definition is built from reusable
1270 *Components (Core Components)* found in the ebXML *Repository*

1271

1272 Each *Party Profile* defines:

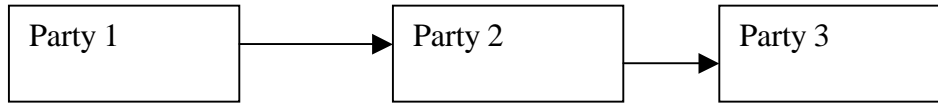
- 1273 • The Commercial Transactions that the *Party* is able to engage into
- 1274 • The Technical *Protocol* (like HTTP, SMTP etc) and the technical properties (such as
1275 special encryption, validation, authentication) that the *Party* supports in the
1276 engagement
- 1277 • The parties acknowledge each other's profile and create a Partner Agreement.
1278 The Partner Agreement references :
 - 1279 • The relevant *Party Profiles*
 - 1280 • The Legal terms and conditions related to the exchange
 - 1281 • The parties implement the respective part of the Profile.
- 1282 This is done:
 - 1283 • Either by creating/configuring a *Business Service Interface*.
 - 1284 • Or properly upgrading the legacy software running at their side
- 1285 In both cases, this step is about :

- 1286 • Plugging the Legacy into the ebXML technical infrastructure as specified by the
- 1287 TR&P
- 1288 • Granting that the software is able to properly engage the stated conversations
- 1289 • Granting that the exchanges semantically conform to the agreed upon Message
- 1290 Definitions
- 1291 • Granting that the exchanges technically conform with the underlying ebXML TR&P
- 1292 • The parties start exchanging messages and performing the agreed upon commercial
- 1293 transactions.
- 1294
- 1295

1295 **Scenario 2: Three or more partners set-up a Business Process**
 1296 **implementing a supply-chain and run the associated exchanges**

1297 The simple case of a supply-chain involving two parties can be reconstructed from the
 1298 Scenario 1.

1299 Here we are dealing with situations where more parties are involved. We consider a
 1300 *Supply Chain* of the following type :



1305
 1306

1307 What fundamentally differs from Scenario 1 is that *Party 2* is engaged at the same time
 1308 with two different parties. The assumption is that the “state” of the entire *Business*
 1309 *Process* is managed by each *Party*, i.e. that each *Party* is fully responsible of the
 1310 Commercial Transaction involving it (*Party 3* only knows about *Party 2*, *Party 2* knows
 1311 about *Party 3* and *Party 1*, *Party 1* knows about *Party 2*).

1312

1313 In this scenario:

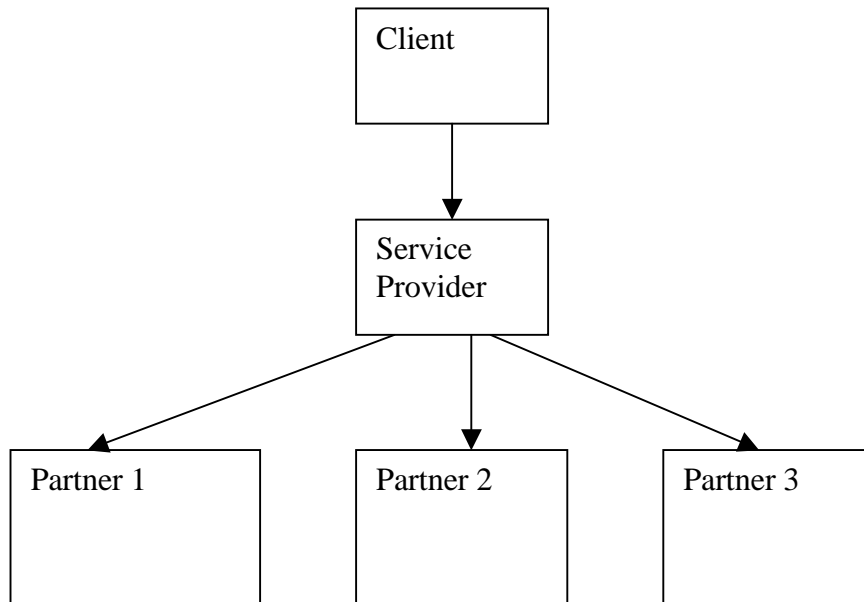
- 1314 • Each partner defines its own *Party Profile*.
- 1315 Each *Party Profile* references:
 - 1316 • One or more existing *Business Process* found in the ebXML *Repository*
 - 1317 • One of more Message Definitions. Each Message definition is built from reusable
 - 1318 *Components (Core Components)* found in the ebXML *Repository*
 - 1319 Each *Party Profile* defines:
 - 1320 • The Commercial Transactions that the *Party* is able to engage into
 - 1321 *Party 2* must be able to support at least 2 Commercial Transactions.
 - 1322 • The Technical *Protocol* (like HTTP, SMTP etc) and the technical properties (such as
 - 1323 special encryption, validation, authentication) that the *Party* supports in the
 - 1324 engagement
 - 1325 the technical requirements for the exchanges with *Party 1* and *Party 3* may be
 - 1326 different. In such case, *Party 2* must be able to support different *Protocols* and/or
 - 1327 properties.
 - 1328 • The parties acknowledge each other profile and create the relevant Partner
 - 1329 Agreements (at least 2 in this Scenario).
 - 1330 Each Partner Agreement references:
 - 1331 • The relevant *Party Profiles*
 - 1332 • The Legal terms and conditions related to the exchange
 - 1333 *Party 2* is engaged in 2 *Party Agreements*.
 - 1334 • The parties implement the respective part of the Profile.
 - 1335 This is done:
 - 1336 • Either by creating/configuring a *Business Service Interface*.
 - 1337 • Or properly upgrading the legacy software running at their side
 - 1338 In both cases, this step is about:

- 1339 • Plugging the Legacy into the ebXML technical infrastructure as specified by the
- 1340 TR&P
- 1341 • Granting that the software is able to properly engage the stated conversations
- 1342 • Granting that the exchanges semantically conform to the agreed upon Message
- 1343 Definitions
- 1344 • Granting that the exchanges technically conform with the underlying ebXML TR&P
- 1345 • *Party 2* may need to implement a complex *Business Service Interface* in order to be
- 1346 able to engage with different partners.
- 1347 • The parties start exchanging messages and performing the agreed upon commercial
- 1348 transactions.
- 1349 • *Party 3* places an order at *Party 2*
- 1350 • *Party 2* (eventually) places an order with *Party 1*
- 1351 • *Party 1* fulfills the order
- 1352 • *Party 2* fulfill the order

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:



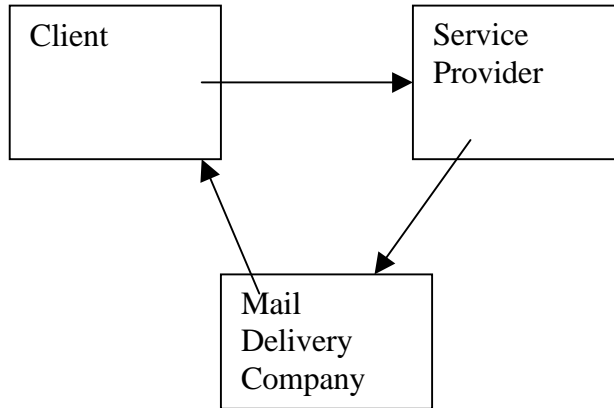
This is an evolution of Scenario 2. The Description of this scenario is omitted.

1377
1378
1379

1379 **Scenario 4: Three or more parties engage in multi-party Business Process**
 1380 **and run the associated exchanges**

1381 This Scenario is about 3 or more Parties having complex relationships. An example of
 1382 this is the use of an external delivery service for delivering goods.

1383
 1384
 1385
 1386
 1387
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395



1396 In this Scenario, each *Party* is involved with more than one other *Party* but the
 1397 relationship is not linear. The good which is ordered by the Client with the Service
 1398 Provider is delivered by a 3rd *Party*.

1399

1400 In this scenario:

- 1401 • Each partner defines its own *Party Profile*.
- 1402 Each *Party Profile* references:
 - 1403 • One or more existing *Business Process* found in the *ebXML Repository*
 - 1404 • One of more *Message Definitions*. Each *Message definition* is built from reusable
 - 1405 *Components (Core Components)* found in the *ebXML Repository*
- 1406 Each *Party Profile* defines:
 - 1407 • The *Commercial Transactions* that the *Party* is able to engage into
 - 1408 In this case, each *Party* must be able to support at least 2 *Commercial Transactions*.
 - 1409 • The *Technical Protocol* (like HTTP, SMTP etc) and the technical properties (such as
 - 1410 special encryption, validation, authentication) that the *Party* supports in the
 - 1411 engagement.
 - 1412 In case the technical infrastructure underlying the different exchanges differs, each
 - 1413 *Party* must be able to support different *Protocols* and/or properties. (an example is
 - 1414 that the order is done through a *Web Site* and the delivery is under the form of an
 - 1415 email).
- 1416 • The parties acknowledge each other profile and create a *Partner Agreement*.
- 1417 Each *Party*, in this Scenario, must be able to negotiate at least 2 *Agreements*.
- 1418 The *Partner Agreement* references :
 - 1419 • The relevant *Party Profiles*
 - 1420 • The *Legal terms and conditions* related to the exchange
 - 1421 Each *Party* is engaged in 2 *Party Agreements*.
 - 1422 • The parties implement the respective part of the *Profile*.
 - 1423 This is done:

- 1424 • Either by creating/configuring a *Business Service Interface*.
- 1425 • Or properly upgrading the legacy software running at their side
- 1426 In both cases, this step is about:
- 1427 • Plugging the Legacy into the ebXML technical infrastructure as specified by the
- 1428 TR&P
- 1429 • Granting that the software is able to properly engage the stated conversations
- 1430 • Granting that the exchanges semantically conform to the agreed upon Message
- 1431 Definitions
- 1432 • Granting that the exchanges technically conform with the underlying ebXML TR&P
- 1433 • All Parties may need to implement complex *Business Service Interfaces* to
- 1434 accommodate the differences in the *Party Agreements* with different Parties.
- 1435 • The parties start exchanging messages and performing the agreed upon commercial
- 1436 transactions.
- 1437 • The Client places an Order at the Service Provider
- 1438 • The Service Provider Acknowledges the Order with The Client
- 1439 • The Service Provider informs the Mail Delivery Service about a good to be delivered
- 1440 at the Client
- 1441 • The Mail Delivery Service delivers the good at the Client
- 1442 • The Clients notifies the Service Provider that the good is received.
- 1443

1443 ***Disclaimer***

1444 The views and specification expressed in this document are those of the authors and are
1445 not necessarily those of their employers. The authors and their employers specifically
1446 disclaim responsibility for any problems arising from correct or incorrect implementation
1447 or use of this design.

1448 ***Copyright Statement***

1449 Copyright © ebXML 2000. All Rights Reserved.

1450

1451 This document and translations of it may be copied and furnished to others, and
1452 derivative works that comment on or otherwise explain it or assist in its implementation
1453 may be prepared, copied, published and distributed, in whole or in part, without
1454 restriction of any kind, provided that the above copyright notice and this paragraph are
1455 included on all such copies and derivative works. However, this document itself may not
1456 be modified in any way, such as by removing the copyright notice or references to the
1457 Internet Society or other Internet organizations, except as needed for the purpose of
1458 developing Internet standards in which case the procedures for copyrights defined in the
1459 Internet Standards process must be followed, or as required to translate it into languages
1460 other than English.

1461

1462 The limited permissions granted above are perpetual and will not be revoked by ebXML
1463 or its successors or assigns.

1464

1465 This document and the information contained herein is provided on an
1466 "AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR
1467 IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE
1468 USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1469 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1470 PARTICULAR PURPOSE.