



Creating A Single Global Electronic Market

1
2
3
4
5
6
7
8
9

Technical Architecture Specification

ebXML Technical Architecture Team

31 August 2000

1.0 Status of this Document

11
12
13
14
15
16
17
18
19
20
21
22

This document specifies an ebXML DRAFT for the eBusiness community. This Document is being distributed for the First Comment Period which shall commence on 20/9/2000 and be closed on 4/10/2000.

The second comment period will commence 11/10/2000 in order to be ready for final approval by the ebXML plenary in Japan.

Distribution of this document is unlimited.

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

23
24
25
26

This version: (0.8.71)

Release Date: 13/9/2000

<http://www.xmlglobal.com/ebXML/downloads>

27
28
29

Latest version:

[http://www.ebxml.org/...](http://www.ebxml.org/)

30
31
32

Previous version:

<http://www.ebxml.org/.....>

33
34
35
36

Comments:

Comments must be sent via email, in plain text format only, to

ebXML-architecture@lists.ebxml.org

37
38
39
40

You must subscribe the list server before you are eligible to post a comment. Before posting comments, please read the archives to avoid duplication of comments.

40

2.0 ebXML participants

41

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

42

43

44

Editor: Duane Nickull, XML Global Technologies

45

Co-Editor: Brian Eisenberg, DataChannel

46

47

48

[NOTE: Add names for TA project team members]

49

49 **Table of Contents**

50

51 **1.0 Status of this Document** 1

52

53 **2.0 ebXML Participants** 2

54

55 **3.0 Introduction** 6

56 3.1 *Summary of Contents of Document* 6

57 3.2 *Audience* 7

58 3.3 *Related Documents* 7

59

60 **4.0 Design Objectives** 8

61 4.1 *Goals* 8

62 4.2 *Caveats and Assumptions* 8

63 4.3 *System Overview* 9

64

65 **5.0 Specifications Roadmap** 10

66

67 **6.0 Architecture Overview** 12

68 6.1 *Introduction and Scope* 12

69 6.2 *Use Case List and Descriptions* 12

70 6.3 *Expanded Run Time View* 14

71 6.4 *Run Time Overview* 15

72 6.5 *Participation Requirements* 19

73 6.6 *General ebXML Design Requirements* 20

74

75 **7.0 Trading Partner Profile Functionality** 21

76 7.1 *Overview* 21

77 7.2 *Requirements* 21

78

79 **8.0 Business Process and Modelling Functionality Requirements** 23

80 8.1 *Overview* 23

81 8.2 *Business Process Documents Requirements* 23

82 8.3 *Business Process Modelling Functional Requirements* 24

83

84 **9.0 Core Component and Common Business Object Functionality** 25

85 9.1 *Overview* 25

86 9.2 *Core Component Functional Requirements* 26

87 9.3 *Common Business Object Functional Requirements* 26

88

89 **10.0 Registry and Repository System** 28

90 10.1 *Overview – Registry and Repository System* 28

91 10.2 *Registry Definition* 28

92 10.3 *Repository Definition* 28

93 10.4 *Functional Requirements* 28

94 10.5 *Requirements for Registry/Repository Authorities* 29

95 10.5.1 *Introduction* 29

96 10.5.2 *Certification* 29

97 10.5.3 *RA Requirements* 29

98 **Table of Contents *cont'd*....**

99

100 10.6 *Repository Items* _____ 30

101 10.7 *Registry and Repository – Distributed vs. Centralized* _____ 30

102 10.7.1 *Introduction* _____ 31

103 10.7.2 *Current Centralized Repository Approach* _____ 32

104 10.7.3 *A New Model for a Distributed Repository* _____ 33

105 10.8 *Registry and Repository Security Requirements* _____ 36

106 10.9 *Addressing Registry/Repository Items* _____ 36

107 10.10 *Querying Registries* _____ 37

108 10.11 *ebXML Decentralized Repository Query Requirements* _____ 37

109

110 **11.0 Messaging and Transport Requirements** _____ **38**

111 11.1 *Introduction* _____ 38

112 11.2 *Messaging Services Description* _____ 38

113 11.3 *Abstract Messaging Services Interface* _____ 39

114 11.4 *Messaging Layer Functions* _____ 40

115 11.5 *Structure and Packaging* _____ 42

116

117 **12.0 ebXML Conformance and Testing** _____ **44**

118 12.1 *ebXML Conformance* _____ 44

119 12.2 *Applicability* _____ 44

120 12.3 *Implementation Conformance* _____ 44

121 12.4 *Application Conformance* _____ 45

122 12.5 *Conformance Inspection* _____ 45

123 12.6 *References* _____ 46

124

125 **Disclaimer** _____ **46**

126

127 **Contact Information** _____ **47**

128

129 **Copyright Statement** _____ **48**

130

131

132 **13.0 ebXML Elaborated Use Case Scenarios** _____ **49**

133

134 **Appendix “A” Elaborated Use Case Descriptions** _____ **50**

135 UC100 *Company Actor Can Register as an SO* _____ 50

136 UC101 *A Trading Partner Actor creates a Trading Partner Profile* _____ 53

137 UC102 *A Trading Partner Actor publishes a Trading Partner Profile to a Registry* _____ 55

138 UC103 *A Registry updates its’ index to reflect changes in a Trading Partner*

139 *Actors’ TPP document* _____ 57

140 UC104 *A Human Actor queries a Registry for a Specific Trading Partners’ TPP* _____ 59

141 UC105 *A Human Actor can query a Registry for a non specific Trading Partner*

142 *Actors’ TPP* _____ 61

143 UC106 *An Application Actor can query a Registry for a Trading Partner Actors’ TPP* _____ 62

144 UC107 *A Human Actor can read a Trading Partner Actors’ TPP* _____ 63

145 UC108 *A Application Actor can parse a Trading Partner Actors’ TPP* _____ 64

146 UC109 *A Trading Partner Actor creates a Business Process in XML syntax* _____ 66

147 UC110 *A Trading Partner Actor publishes a Business Process to a Registry* _____ 66

148 UC111 *A Human Actor can identify a Business Process supported by a Trading*

149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182

Table of Contents *cont'd....*

	Partner Actor	68
	UC112 An Application Actor can identify a Business Process from its' GUID (Globally Unique Identifier)	68
	UC113 An Application Actor can identify a DTD (Schema) based on its' GUID	68
	UC114 An Application Actor can identify a Core Component based on its' GUID	68
	UC115 A Human Actor can review a copy of a Repository Item in a GUID	68
	UC117 An Application Actor can query the Repository for a Core Component	69
	UC118 A Human Actor can Query the Repository for a Repository Item on criteria other than GUID	70
	UC119 Trading Partner Actors ebXML Application resolves a semantic equivalency or a Core Component (or equivalent) referenced in a business message instance	70
	UC120 A Human Actor can submit a concatenation or modification to a Core Component to a RA (Repository Authority)	71
	UC121 A Company Actors ebXML Business Application Constructs and exchanges a series of messages to complete a business transaction with another Company Actor	71
	UC122 A Trading Partner Actor creates a directory Structure and configures the ebXMLind.xml file	71
	UC123 A semantic Mapping Specialist Actor can review a Registry submission	71
	UC124 A Registry/Repository Authority can Modify a Core Component	71
	UC125 The Repository can accept the Queries in an ebXML compliant message	71
	UC126 A Registry Actor communicates with another Registry Actor	71
	UC127 The Registry/Repository communicates an exception error to the RA.	71
	UC128 A Human Actor models a business process in UML.	71
	Appendix "B" - Implementation Issues	72
	Appendix "C" - Examples of Documents	76

182

3.0 Introduction

183

184 Although XML is a recent newcomer in the electronic commerce landscape, supply
185 chains in many industries, as well as industry consortiums and standards organizations
186 are using XML to define their own vocabularies for business relationships and
187 transactions. The vocabularies, business templates, and business processes used by these
188 groups to transact business must be accessible by all partners at any time. Furthermore,
189 newcomers to the supply chain or business partnerships must be able to discover these
190 interfaces and be able to interoperate between them in a secure, reliable and consistent
191 manner. In order to facilitate these needs, a mechanism must be in place to be able to
192 provide information about each participant *Trading Partner Actor* including what they
193 support for business processes and interfaces, information about what business
194 information is required for each instance of a business message and a mechanism to allow
195 dynamic discovery of the semantic meaning of that business information. The entire
196 mechanism must be able to recognize semantic meaning at the XML element level,
197 moving away from a document centric world. A registry/repository based system can be
198 used to provide this functionality. A series of registries and distributed repositories,
199 accessible via a consistent method within the infrastructure, can link many organizations
200 and industries, acting as a web of registries for discovery. Standards are needed to ensure
201 interoperability of these registries; additionally, a registry bound open vocabulary of core
202 components must be created for consistency of discovery among proprietary
203 vocabularies.

204

3.1 Summary of Contents of Document

205

206 The ebXML Technical Architecture Specification represents the work of the ebXML
207 Technical Architecture Project Team. This document is based on the set of requirements
208 outlined in the ebXML Requirements Specification [Insert: URL link]. It provides a high
209 level abstraction of the underlying architecture for ebXML, the relationships between
210 each of the seven major component specifications, requirements for participation, and
211 ebXML conformance and testing.

212

213 It also specifies what each of the subsequent specifications must facilitate in terms of
214 functionality and conformance. The Registry and Repository, Transport / Routing and
215 Packaging, Business Process Modelling and Core Component specification must conform
216 to the functional requirements specified hereinafter. One of the underlying concepts of
217 ebXML is extensibility hence an implied ability to extend the basic functionality beyond
218 what is specified herein.

219

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

223 **3.2 Audience**

224 The target audience for this specification includes the ebXML project teams, software
225 developers, international standards organizations, and industry groups.

226 **3.3 Related Documents**

227 The following documents provide detailed definitions for each component specification.
228 They include ebXML Specifications on the following topics:

229

- 230 1. Trading Partner Profile Specification
- 231 2. Business Process Specification
- 232 3. Core Components Specification
- 233 4. Registry Repository [Link to RegRep spec]
- 234 5. Messaging Services [Link to TRP spec]
- 235 6. Glossary of Terms [Link to ebXML glossary]

236

237 The following documents are also relevant to the ebXML infrastructure:

238

- 239 1. XML v 1.0 specification (www.w3c.org/xml/TR/021998.html)
- 240 2. ISO 11179 Metadata Repository [get exact info]

241

242

242

4.0 Design Objectives

243 **4.1 Goals**

244

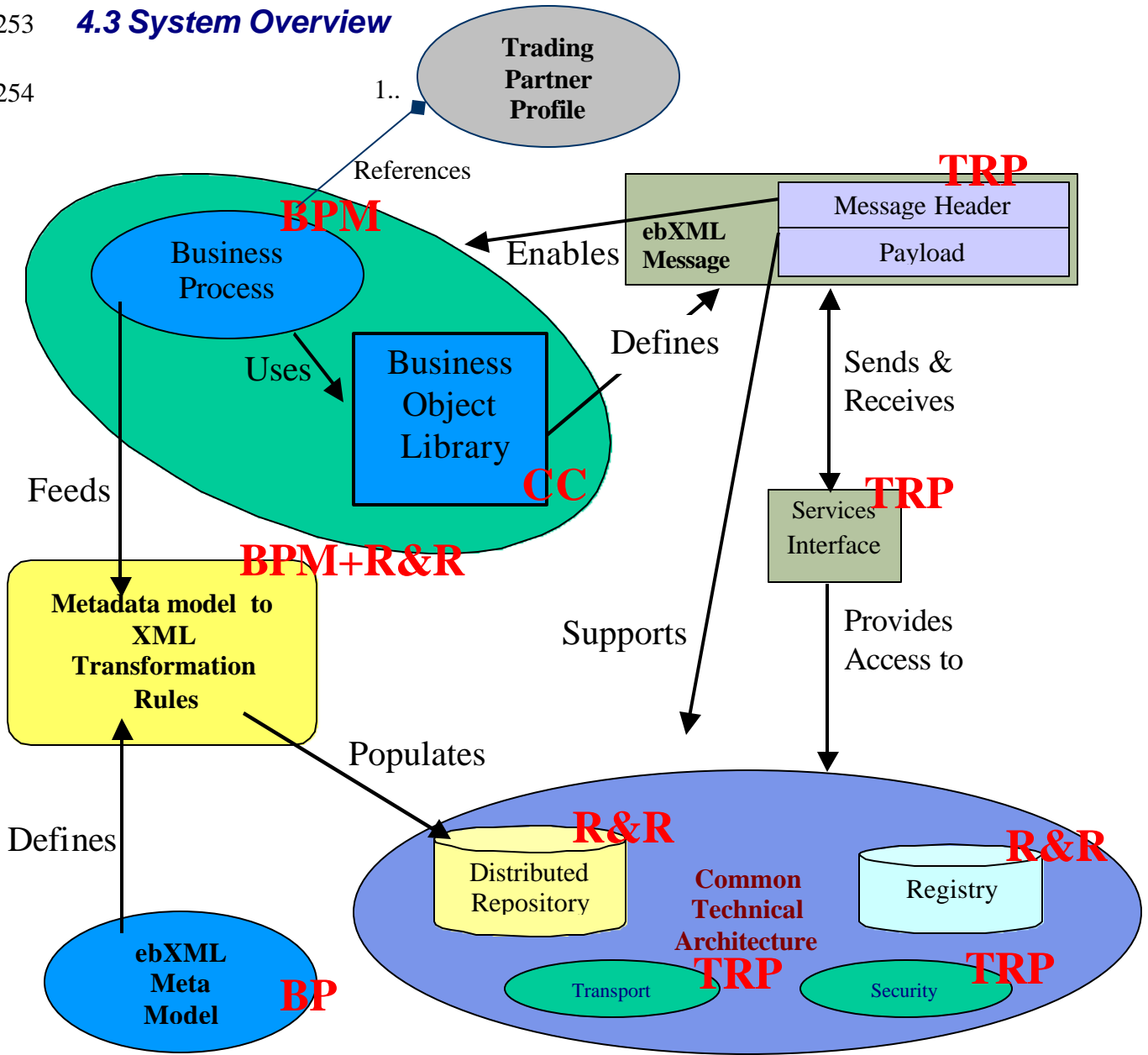
245 This specification, in conjunction with the ebXML Requirements Specification, shall be
246 used by all ebXML project teams in determining the areas of responsibility in developing
247 the ebXML 1.0 Standard, and shall guide their efforts in developing the technical
248 specifications necessary to satisfy their respective requirements.

249 **4.2 Caveats and Assumptions**

250 This specification is a high-level roadmap that provides a description of the underlying
251 architecture for ebXML. It describes each component specification at a high level of
252 abstraction and outlines the interrelationships between them.

253 **4.3 System Overview**

254



255
256
257
258
259
260
261
262
263
264

Figure 1.0 Represents an abstraction of business and technological components that are within the scope of the ebXML Technical Architecture. The inter-relationships between components define the mandate for interoperability. The red characters represent the ebXML project team primarily responsible for each area: **BPM** = Business Process Methodology, **TRP** = Transport, Routing & Packaging, **R&R** = Registry and Repository, **CC** = Core Components.

264

265 **5.0 Specifications Roadmap**

266

267 The ebXML 1.0 specification shall be expressed as a series of design rules and technical
268 specifications. There are six major component specifications that collectively form the
269 final specification. Each of these six major specifications MAY, in turn, include other
270 sub-specifications.

271

272

273

- 274 1. **Technical Architecture (TA)** - contains an overview of the technical
275 infrastructure that comprises ebXML and itemizes the design rules and
276 conformance guidelines. It also provides a minimal set of functionality and
277 performance requirements for each of the subsequent specification to facilitate
278 and a set of conformance tests for ensuring that all application Actors are able to
279 interoperate in a manner consistent with the design principles of ebXML.

280

- 281 2. **Trading Partner Profile (TPP)** – elaborates the functionality and syntax
282 requirements for stating Trading Partner Profile information in XML format that
283 complies with the ebXML infrastructure and TA conformance tests.

284

285

- 286 3. **Business Process and Modelling Specifications (BPM)** - the business process
287 Specification SHALL include rules for modelling and deriving business processes
288 into an XML syntax and the subsequent expression of the processes in XML
289 format that complies with the ebXML infrastructure and TA conformance tests.

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

- 308 5. **Registry and Repository (RR)** - includes functional specification and technical
309 design, interfaces, services for constructing an ebXML compliant
310 Registry/Repository System which must conform to a minimum set of
311 performance criteria. The minimum performance and functionality criteria is set
312 forth in this Technical Architecture Specification.

308

309

6. **Messaging Services (TRP)** - The ebXML Messaging Specification provides a set of requirements for building ebXML conformant messages. The ebXML messaging format facilitates reliable and secure delivery of ebXML messages over various underlying transport protocol(s). It supports simplex (one-way) and request/response (either synchronous or asynchronous) message exchange. It also provides security, i.e., authentication, access-control, integrity, non-repudiation, and privacy, and enforces the rules of the Trading Partner Profile (TPP).

310

311

312

313

314

315

316

317

318

318

6.0 Architecture Overview

319

6.1 Introduction and Scope:

321

322 This section of the Specification document is an overview of the architectural
 323 components of the ebXML infrastructure. The Architecture is expressed in a series of
 324 Run Time views, Use Cases, and System/Architectural overviews. The overviews
 325 provide a clear picture of what each component must facilitate in order to comply with
 326 the ebXML Requirements Specification. The functionality is considered a base set of
 327 minimal requirements and it is implicitly understood that individual software vendors
 328 may choose to extend the functionality of their system as long as it does not override this
 329 basic functionality. Additionally, all components must be able to stand up to a series of
 330 Conformant tests (see section 12 - “ebXML Conformance and Testing”).
 331

6.2 Use case list and descriptions

333

334 The Use Case list below was identified during the development of this architecture
 335 Specification. The following Use Case list is elaborated upon at the end of this
 336 Specification (see Appendix A – “**Elaborated ebXML Use Case Scenarios**”).
 337

338

Use Cases for overall core ebXML functionality:

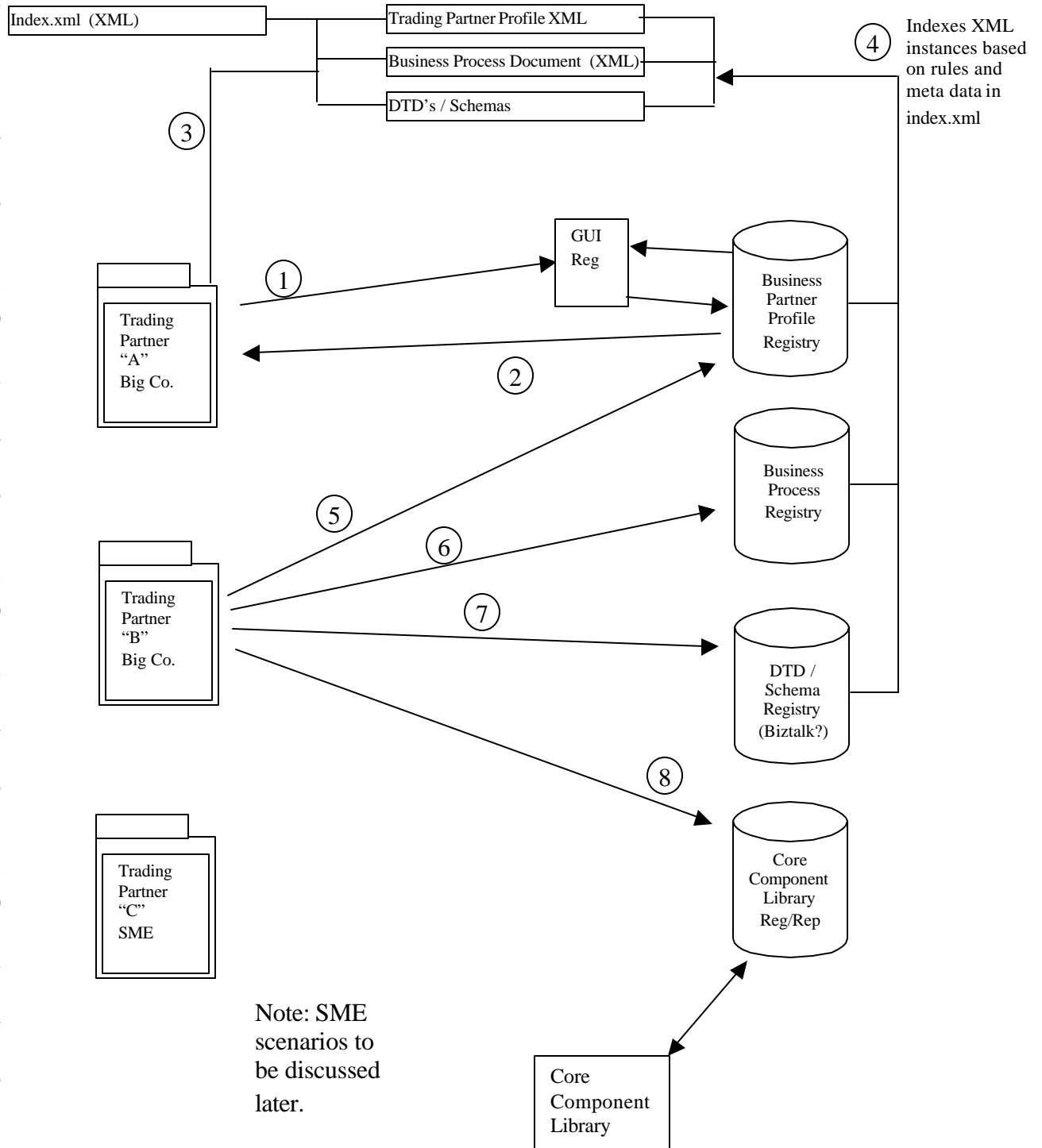
339

- 340 **UC100** A Company Actor can Register as an SO (Submitting Organization [participant])
- 341 **UC101** A Trading Partner Actor creates a Trading Partner Profile
- 342 **UC102** A Trading Partner Actor publishes a Trading Partner Profile to a Registry
- 343 **UC103** A Registry updates its' index to reflect changes in a Trading Partner Actors' TPP document
- 344 **UC104** A Human Actor queries a Registry for a Specific Trading Partners' TPP
- 345 **UC105** A Human Actor can query a Registry for a non specific Trading Partner Actors' TPP
- 346 **UC106** An Application Actor can query a Registry for a Trading Partner Actors' TPP
- 347 **UC107** A Human Actor can read a Trading Partner Actors' TPP
- 348 **UC108** A Application Actor can parse a Trading Partner Actors' TPP
- 349 **UC109** A Trading Partner Actor creates a Business Process in XML syntax.
- 350 **UC110** A Trading Partner Actor publishes a Business Process to a Registry
- 351 **UC111** A Human Actor can identify a Business Process supported by a Trading Partner Actor
- 352 **UC112** An Application Actor can identify a Business Process from its' GUID (Globally Unique Identifier)
- 353 **UC113** An Application Actor can identify a DTD (Schema) based on its' GUID
- 354 **UC114** An Application Actor can identify a Core Component based on its' GUID
- 355 **UC115** A Human Actor can review a copy of a Repository Item in a GUI
- 356 **UC117** An Application Actor can query the Repository for a Core Component
- 357 **UC118** A Human Actor can Query the Repository for a Repository Item on criteria other than GUID
- 358 **UC119** Trading Partner Actors ebXML Application resolves a semantic equivalency or a Core Component
 359 (or equivalent) referenced in a business message instance
- 360 **UC120** A Human Actor can submit a concatenation or modification to a Core Component to a RA
 361 (Repository Authority)

- 362 **UC121** A Company Actors ebXML Business Application Constructs and exchanges a series of messages
363 to complete a business transaction with another Company Actor
- 364 **UC122** A Trading Partner Actor creates a directory Structure and configures the ebXMLind.xml file.
- 365 **UC123** A semantic Mapping Specialist Actor can review a Registry submission
- 366 **UC124** A Registry/Repository Authority can Modify a Core Component
- 367 **UC125** The Repository can accept the Queries in an ebXML compliant message
- 368 **UC126** A Registry Actor communicates with another Registry Actor
- 369 **UC127** The Registry/Repository communicates an exception error to the RA.
- 370 **UC128** A Human Actor models a business process in UML.
- 371

6.3 Expanded Run Time view (see notes - section 6.4)

371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411



412

413 ***Notes to Accompany Figure 6.4:***

414

415 Each Registry on the right hand side represents an abstract layer of a Registry. Each
416 component does not necessarily have to be a different Registry however, it is possible
417 that may happen at implementation time.

418

419 The Registry labelled “DTD/Schema Registry” will most likely contain only DTD’s. The
420 entire infrastructure has been architected to be agonistic to the syntax of the associated
421 meta data mechanism. This means that schemas MAY be used in place of DTD’s
422 although this is dependant on the finalization of the W3C’s XSDL. The ebXML
423 architecture components MUST use validating parsers in order to facilitate the desired
424 functionality. For schema Repositories to interoperate within the ebXML infrastructure,
425 such the BizTalk™ Schema Repository and the OASIS XML.org Schema Repository,
426 they must contain the Globally Unique Identifiers (GUID) for each element to reference
427 an associated metadata file as described herein. Participating Registry/Repositories MAY
428 also be required to accept ebXML compliant messages.

429

430 The Core Component Library SHOULD NOT be submitted by Trading Partners. It is
431 developed as a starter set of data elements for building business messages. If a Trading
432 Partner wants to add a data element and publish it for use by others, it must submit it to a
433 Repository Authority of an ebXML compliant Registry/Repository who will review the
434 submission. Upon approval, the newly created Data Element Item shall be expressed in
435 XML syntax that conforms to the ebXML specifications and in compliance with the
436 ebXML Registry and Repository specification.

437

438 The Process for extending a Core Component and submitting it as a new Data Element to
439 a RA will be determined by the Registry and Repository Project Team and the Core
440 Component Group.

441

442

443 ***6.4 Run Time Overview***444 ***(Based on Figure 6.3 - above)***

445 (1) A Trading Partner SHALL be able to access an Interface which opens a form
446 allowing the Trading Partner to register as a Submitting Organization (note: the term SO
447 is used to describe the Role of the Actor labeled Trading Partner)

448 (2) The Trading Partner SHALL receive information that describes how to publish the
449 following information. This MAY be accomplished by uploading certain documents

Technical Architecture Specification

Page 15 of 83

450 (such as a TPP document) to a standardized directory structure on an Internet Accessible
451 server OR storing such documents in a Centralized Repository OR a combination of both.

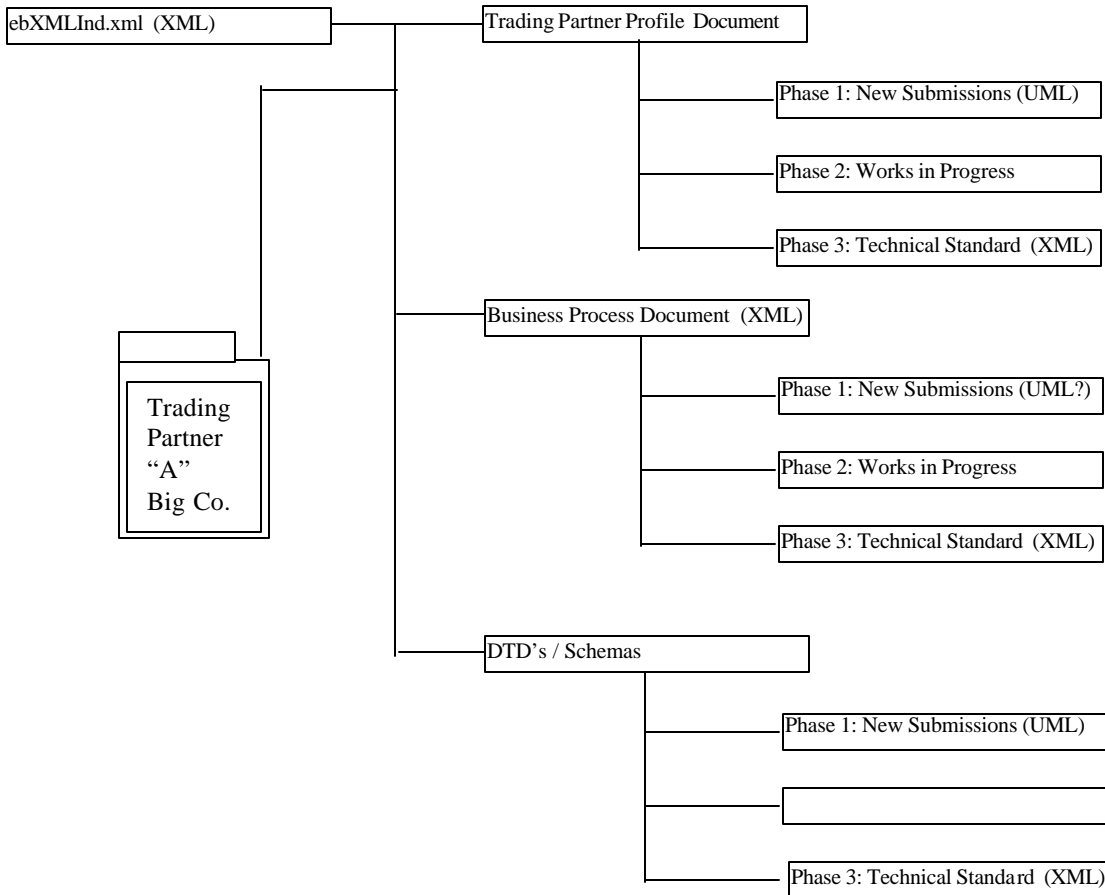
452 a) A Trading Partner Profile (see section 7.0 “**Trading Partner Profile**
453 **Functionality**”). This TPP MUST contain a list of all Business Processes
454 (see section 8.0 “**Business Process Requirements**”) supported by the Trading
455 Partner Actor. The Business Processes identified in the TPP MUST contain
456 Globally Unique Identifiers which reference a file in an ebXML compliant
457 Business Process Registry.

458 b) Copies of all supported business processes (see section 8.0 “**Business Process**
459 **Requirements**”) they wish to submit to an ebXML compliant Registry. A
460 Trading Partner Actor MAY chose to use existing Business processes which
461 are referenced in an ebXML compliant Registry rather than creating and
462 publishing their own. The Business Processes MUST contain References (via
463 GUIDs – “Globally Unique Identifiers”) to specific DTD’s and/or Schemas in
464 an ebXML compliant DTD/Schema Registry.

465 c) Copies of all DTD’s/Schemas referenced in each Business Process which are
466 to be submitted to an ebXML compliant DTD/Schema Registry. The DTD’s
467 MUST contain references from each ELEMENT to specific atomic data
468 elements (see section 9.0 “**Core Component and Common Business Object**
469 **Functionality**”). A Trading Partner Actor MAY chose to use existing DTD’s
470 and/or Schemas which can be referenced in an ebXML compliant Registry
471 rather than creating and publishing their own. The DTD’s and/or Schemas
472 MUST contain References (via GUIDs) to specific Core Components, or an
473 extension of the Core Component Library as described in section 9.0, in an
474 ebXML compliant DTD/Schema Registry.

475
476

476 (3) The Trading Partner MUST publish the following business information to a Registry.
 477
 478



493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504 *Figure 6.5 – Static view of Business Information*
 505

506 A Registry Agent MUST be able access the business information documents that the
 507 Trading Partner Actor wishes to publish to a Registry. A Registry Agent shall have
 508 access to those files for purposes of indexing them. IN cases of a non centralized
 509 Repository, the Registry Agent shall utilize a special valid xml file named
 510 ebXMLInd.xml (see Appendix B - “Sample ebXMLInd.xml file”) which a Registry
 511 Agent MAY use as a method to determine which items are to be included in the
 512 Registry’s index. A Registry Authority shall publish the necessary explanations of how
 513 to publish and change a special file called “ebXMLInd.xml”. This file SHALL publish
 514 meta data related to new additions or changes to existing items being indexed by any
 515 ebXML compliant registry. The Registry MUST force its’ agents to parse this file before
 516 indexing any of the documents described above in section 2(a-c). See section 10.0
 517 “**Registry and Repository Functional Requirements and Overview**” for more
 518 information on publishing and submitting items in compliance with the policies and rules
 519 of the Registry.
 520

521 The procedure MAY be different in cases of a centralized Repository being deployed
522 and physically containing the documents.
523

524 All the documents and data outlined in this section MAY be published by the Trading
525 Partner Actor using a web server.
526

527 (4) The Registrys' Agent MUST parse the special file named "ebXMLind.xml" before
528 sending an agent to the Trading Partners' Directory to retrieve copies of files for indexing
529 by the Registry. This is to ensure only content which is ready for submitting to a
530 Registry gets included in a Registry Index.
531

532 (5) A Trading Partner Actor SHALL be able to query the Business Profile Registry
533 Interface in adherence with the rules and policies of the ebXML **compliant** Registry. A
534 successful Query result MUST contain a location where they can retrieve a TPP
535 document. They MUST be able to retrieve it by issuing a 'get' type request to the
536 location. The Trading Partner Profile (TPP) MUST contain business information about
537 which business processes are supported by the Trading Partner along with other
538 information required and outlined in the Trading Partner Profile Specification [NOTE-
539 provide link to TPP spec here].
540

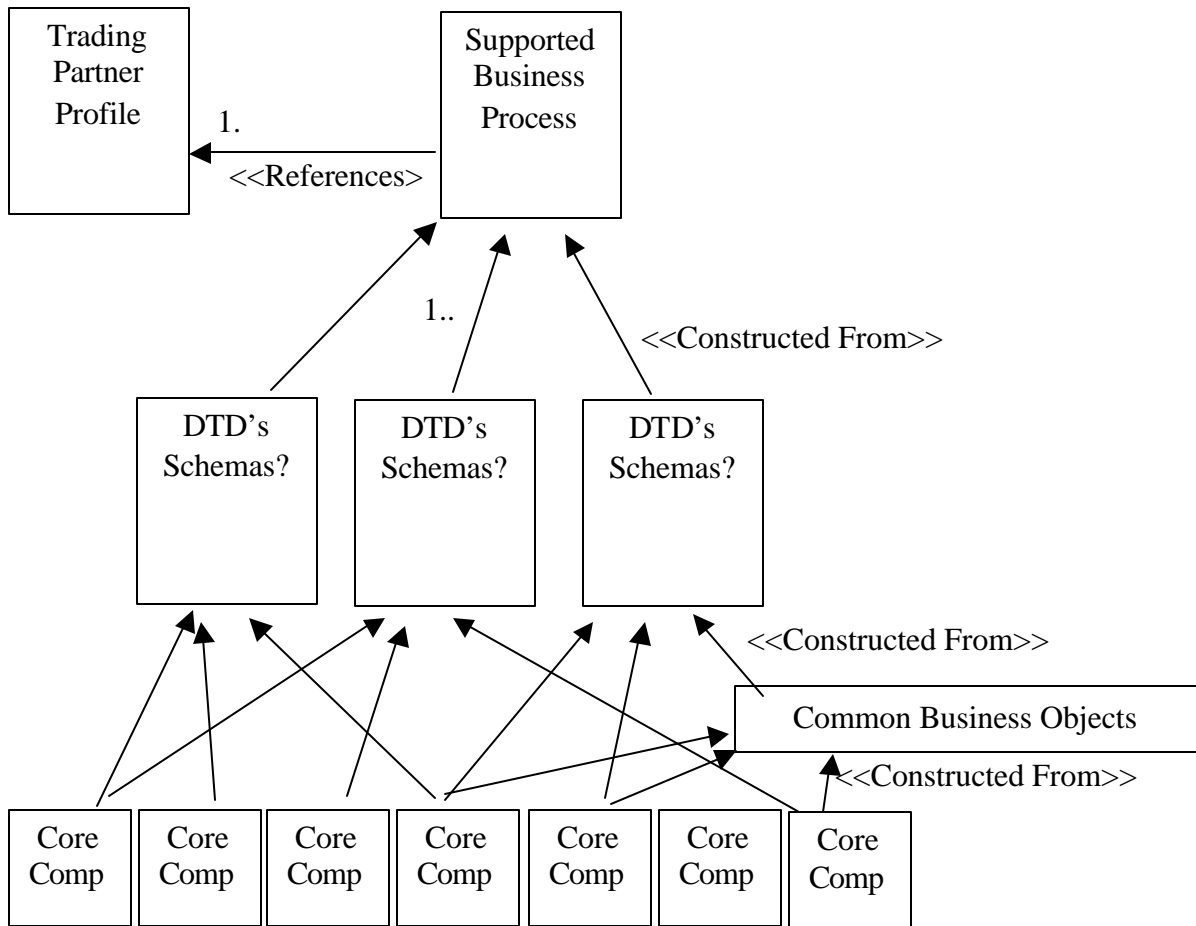
541 (6) A Trading Partner Actor SHALL be able to query the Business Process Registry
542 Interface in adherence with the rules and policies of the ebXML **compliant** Registry. A
543 successful Query result MUST contain a location where they can retrieve a Business
544 Process document. They MUST be able to retrieve it by issuing a 'get' type request to
545 the location. The Business Process Document MUST contain business information
546 which describes a business process including, but not limited to, the Roles of all
547 participants, the sequence of events and references to the business information instances
548 required for meeting the Trading Partner Actors business requirements as defined by the
549 Business Process Specification.
550 [NOTE- provide link to BPM spec here].
551

552 The Business Process document instances MUST contain globally unique identifiers to
553 DTD's or Schemas which can be (7) retrieved and examined, possible as a result of a
554 query to an ebXML **compliant** Registry. The DTD's/Schemas represent business
555 message document instances that are part of a Business Process. If all the DTD's or
556 Schemas which comprise a specific Business Process are known to all Trading Partner
557 Actors (as will likely be the case within vertical organizations), the Business Process can
558 be commenced.
559

560 Each Element in a DTD or Schema MUST have a reference to a Meta Data item in an
561 ebXML **compliant** Registry/Repository. This reference MUST be accomplished by
562 relating a Globally Unique Identifier to each data element using an appropriate
563 methodology available in the XML v 1.0 (or superceded) specification. In the case of
564 DTD's, this SHALL be accomplished by utilizing a fixed attribute value for each XML

565 element. (see section 10.0 – “**Registry and Repository Functional Requirements and**
 566 **Overview**” for a full description of this mechanism.)

567
 568 If the DTD’s are not recognized, the Trading Partner Actor MAY still dynamically
 569 construct it based on resolving each elements’ fixed attribute to a special XML file
 570 known as a “Core Component” and/or a series of Common Business Objects built with
 571 Core Components or a proprietary Registry XML element item which is constructed and
 572 published in accordance with the ebXML specifications and methodologies.



602 *Figure 6.6 – Relationship view of ebXML Business Information*

604 **6.6 Participation Requirements**

607 At the very least, all ebXML Trading Partner Actors MUST:

- 609 A) Register with a Repository Authority as a Submitting Organization.
610
611 B) Publish a Trading Partner Profile in an ebXML compliant Registry which MUST
612 contain a minimal amount of information to facilitate business Interchanges as
613 defined in the Trading Partner Profile Specification.
614
615 C) Publish or link (from their Trading Partner Profile document) at least one
616 Business Process which they support.
617
618 D) Have a web addressable system available for a business interface to allow other
619 Trading Partner Actors the ability to engage in Business Processes published in
620 the Trading Partner Profile.
621
622 E) Use ebXML Compliant and Conformant components as specified in Section 12.0
623 herein.
624
625

626 SMEs (Small to Medium sized Enterprises) MAY be able to participate in ebXML by
627 using web based service packages and accessing all functionality via a web browser.
628

629 Large corporations MAY elect to participate by building or purchasing their own
630 enterprise level ebXML Application server.
631
632

633 **6.7 General ebXML Design Requirements**

634
635 participating components in ebXML MUST facilitate multilingual support. ebXML
636 specification MUST be compliant with Unicode and ISO/IEC 10646 for character set,
637 and UTF-8 or UTF-16 for character encoding.
638

639 There MUST be no single point of failure in the ebXML architecture and infrastructure.
640 This specifically refers to registry and repository failure.
641
642
643
644
645
646
647
648
649
650
651
652

653
654
655

656 **7.0 Trading Partner Profile Functionality**

657

658 **7.1 Overview**

659

660 A Trading Partner Profile is a document which adheres to a DTD specified in the Trading
661 Partner Profile Specification [link here]. The Trading Partner Profile document (XML
662 Syntax) SHALL facilitate a Trading Partner Actor being able to express it's minimal
663 legal and business requirements in a manner where it can be universally understood by
664 other ebXML compliant Trading Partner Actors. The document must be indexed by at
665 least one ebXML compliant Registry in order to be accessible by other ebXML Trading
666 Partner Actors.

667

668 For an Example of a Trading Partner Profile Agreement, please see the Trading Partner
669 Profile Specification OR Appendix "C" Example Documents contained herein.

670 **7.2 Functional Requirements**

671

672 **7.2.1** The following Requirements MUST be met by the Trading Partner profile:

673

- 674 a) The TPP SHALL be expressed as a valid XML document.
- 675
- 676 b) The TPP SHALL contain legal and contact information in a human readable form
- 677
- 678 c) The TPP SHALL contain one or more occurrences of Business Processes which
- 679 are supported by the Trading Partner Actor. Each instances of a Business Process
- 680 MUST be indexed in an ebXML compliant Registry and identifiable by a
- 681 Globally Unique Identifier (GUID)/
- 682
- 683 d) The TPP SHALL identify at least one human contact for a Trading Partner Actor
- 684
- 685 e) The TPP SHALL contain sufficient information to meet the legal requirements of
- 686 businesses.
- 687
- 688 f) The TPP SHALL contain sufficient information to make known the Trading
- 689 Partners Communication, Security and Web address information.
- 690
- 691 g) The TPP SHALL contain other information to be determined by the ebXML
- 692 Trading Partner Profile Project Team and expressed in the resultant Specification.
- 693

694 **7.2.2** The TPP for each ebXML compliant Trading Partner **MUST** be indexed by one or
695 more ebXML compliant Registries.

696

697 **7.2.3** A TPP **MUST** be retrievable at run time by an ebXML Trading Partner Actor by the
698 methodologies described in the ebXML Transport, Routing and Messaging Specification.

699

700

701

702

702 **8.0 Business Process and Modelling Functionality** 703 **Requirements**

704

705 **8.1 Overview**

706

707 A Business Process Document is a roadmap for a specific business transaction. A
708 Business Process document SHALL be expressed in a valid XML document which
709 adheres to a DTD specified in the Business Process and Modeling Specification [link
710 here]. The functionality of the Business Process document (XML Syntax) SHALL
711 facilitate a Trading Partner Actor being able to define a real world business transaction
712 which can be consistently interpreted by other ebXML compliant trading partner Actors.
713 A Business Process document must be indexed by at least one ebXML compliant
714 Registry in order to be accessible and readable by other ebXML Trading Partner Actors.

715

716 For an Example of a Business Process Document, please see the BPM Specification OR
717 Appendix “C” – Example Documents contained herein.

718

719 The modeling of a Business Process MAY be done in a manner which is consistent with
720 the methodologies outlined in the Business Process Modeling Specification. At run time,
721 Trading Partner Actors and their Business Application Actors SHALL utilize only the
722 XML syntax documents, specifically not any UML models.

723 **8.2 Business Process Documents Requirements**

724

725 The following Components and functionality MUST be present in the Business Process
726 Documents (BPD):

727

- 728 a) The BPD used at runtime MUST be expressed as a valid XML document.
- 729
- 730 b) The BPD SHALL contain a Role Matrix for each party to the Process.
- 731
- 732 c) The BPD SHALL contain one or more occurrences of steps which are necessary
733 to complete the Business Transaction. Each step MAY require a Trading Partner
734 Actor to send a specific set of business information to another Trading Partner
735 Actor. Each set of Business information MUST be specified by a DTD or
736 Schema that is indexed by one or more ebXML compliant Registries and can be
737 queried for and retrieved by its' Globally Unique Identification (GUID).
- 738
- 739 d) The BPD shall contain information to recover from any errors during the
740 transaction process such as, but not limited to, server time out limits being
741 exceeded, security protocols not being present or corrupt data. The Recovery
742 MAY include a step to report such errors to an Application Actors' log file or a
743 Human Actor. (see also Transport, Routing and Packaging).

744

745

746

747

Each Business Process Document **MUST** be indexed in one or more ebXML compliant Registries and identifiable by a Globally Unique Identifier (GUID) and retrievable by other ebXML Trading Partner Actors in adherence with the Business Rules of the Trading Partner.

748

749

750

751

8.3 Business process Modeling Functional Requirements

752

753

754

755

756

757

Business Process Modeling **SHALL** not be deemed a minimal requirement for Trading Partner Actor participation in ebXML. Trading Partner Actors, particularly those known as SME's, **MAY** alternatively opt to adopt packages of standard Business Processes already expressed in XML syntax in accordance with the ebXML series of Specifications.

758

759

760

761

762

a. A Method **SHALL** be specified (within the “**Business Process and Modelling Specification**”) for those Trading Partner Actors who wish to model their Business Processes and derive ebXML compliant XML syntax documents in a consistent and reliable manner.

763

764

765

b. UML (the OMG “**Unified Modelling Language**”) **MAY** be used to model Business Processes.

766

767

768

769

770

c. When the model is translated into an XML syntax document and before it is indexed by a Registry, it **MUST** conform to a DTD which **MUST** contain the minimal information required for ebXML functionality (defined in the “**Business Process and Modelling Specification**”)

771

772

773

774

775

The third phase of a Repository Item life cycle **MUST** be expressed in Valid XML syntax. The OMG (Object Management Group) XMI (Xml Metadata Interchange) methodology **MAY** be used for transition of a Repository Item from a UML diagram to Valid XML syntax. The Requirements for the transformation methodology are as follows:

776

777

778

779

780

781

782

783

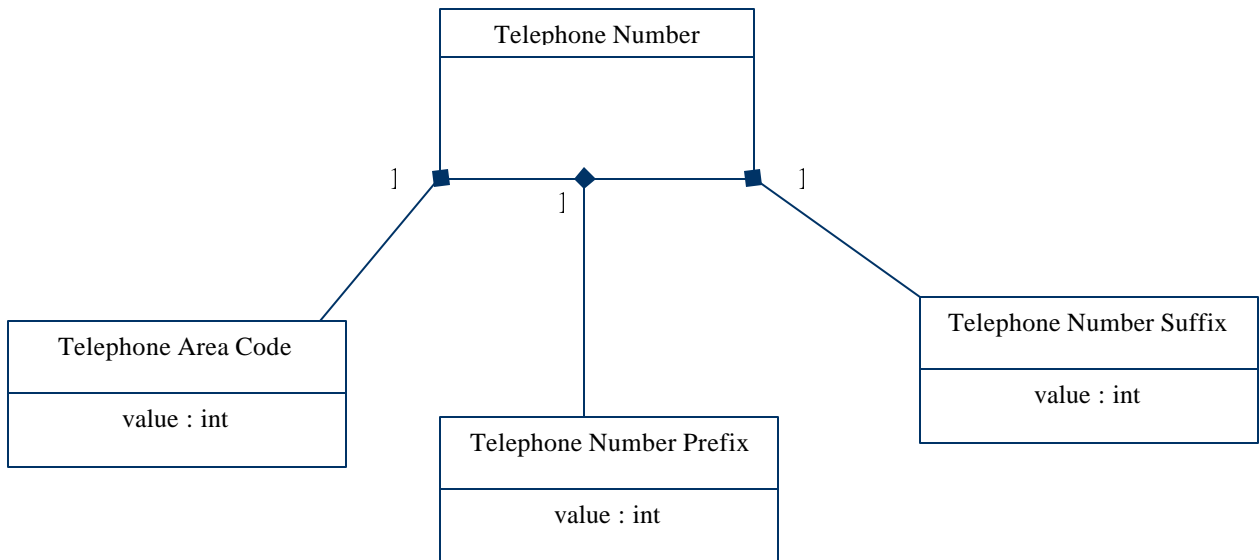
784

a) The methodology used **MUST** yield consistent results

b) The methodology **MUST** use a stable, open and published standard for such transformations

800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824

Example of a Common Business Object). Examples of Core Components may be “Telephone area code”, “Telephone number prefix” and “Telephone number suffix”. Together, they MAY form an example of a Common Business Object called “Telephone Number” which is accepted by many different Trading Partner Actors in different forms.



825 **Example X.X** – A Class view Diagram of a *Common Business Object* called “Telephone
826 Number” and its’ decomposition into *Core Components* called “Telephone area code”,
827 “Telephone number prefix” and “Telephone number suffix”.
828 For an Example of a Core Component Document, please see the Core Component
829 Specification OR Appendix “C” – Example Documents contained herein.
830

831 **9.2 Core Component Functional Requirements**

832
833 The ebXML Core Component Project Team SHALL define a starter set of Core
834 Components which shall reside in a Registry/Repository system (see section 10.0 –
835 “**Registry/Repository Functional Requirements**”)
836

837 The EbXML Core Component library SHALL contain enough Core Components to
838 allow construction of most common business messages.
839

840 The ebXML Core Components SHALL be expressed as Valid XML documents in
841 adherence to the DTD specified in the Core Component Specification.
842

843 The ebXML Core Component Library SHALL be deemed “Open Source” or equivalent
844 in order that anyone may copy, use and recognize the Core Component Library.
845

846 The ebXML Core Component Library SHALL be kept in a Special Registry/Repository
847 yet MAY be mirrored by other ebXML compliant Registry/Repository Systems in order
848 to facilitate scalability and mitigate the possibility of a single point of failure in the
849 infrastructure.
850

851 Each of the ebXML Core Components SHALL contain enough metadata information to
852 convey the semantic meaning of the Component in adherence to the Core Component
853 Specification.
854

855 The ebXML Core Component DTD MUST provide a mechanism for identification of
856 semantic equivalencies to data Elements in other taxonomies.
857
858

859 **9.3 Common Business Object Functional Requirements**

860
861 For an Example of a Common Business Object, please see the Core Component
862 Specification OR Appendix “C” – Example Documents contained herein.
863
864

865 The ebXML Core Component Project Team SHALL define a starter set of Common
866 Business Objects which shall reside in a Registry/Repository system (see section 10.0 –
867 “**Registry/Repository Functional Requirements**”)

868

869 The EbXML Common Business Object library SHALL use only the ebXML Core
870 Components as described herein.

871

872 The ebXML Common Business Library SHALL be expressed as Valid XML documents
873 in adherence to the DTD specified in the Core Component Specification.

874

875 The ebXML Common Business Library SHALL be deemed “Open Source” or equivalent
876 in order that anyone may copy, use and recognize the Common Business Object Library.

877

878 The ebXML Common Business Object Library SHALL be kept in a Special
879 Registry/Repository operated by OASIS yet MAY be mirrored by other ebXML
880 compliant Registry/Repository Systems in order to facilitate scalability and mitigate the
881 possibility of a single point of failure in the infrastructure.

882

883 Each of the ebXML Common Business Library SHALL contain enough metadata
884 information to convey the semantic meaning of the Objects in adherence to the Core
885 Component Specification.

886

887 The ebXML Common Business Object format DTD MUST provide a mechanism for
888 identification of semantic equivalencies to Common Business Objects in other
889 taxonomies.

890

891

892

892

893

10.0 Registry and Repository Functional Requirements and Overview

894

895

896

10.1 Overview - Registry and Repository System

898

899 A Registry/Repository system MAY have many deployments models which yield the
900 same functionality. The goal of the Technical Architecture Specification is to define the
901 minimal functional requirements which a Registry/Repository System MUST provide to
902 facilitate its role in the ebXML infrastructure and outline the logical architecture view for
903 deployment of such systems. The Registry/Repository Project Team SHALL produce the
904 final specification or set of specifications which constrain the deployment and functional
905 requirements of a Registry and Repository mechanism.

906

10.2 Registry Definition

908

- 909 a. A *registry* is a mechanism whereby relevant repository items and metadata about
910 them can be registered such that a pointer to their location, and all their metadata, can
911 be retrieved as the result of a query.

912

10.3 Repository Definition

914

- 915 a. A repository is a location or a set of distributed locations where *Repository Items*
916 pointed at by the registry reside and from which they can be retrieved by any ebXML
917 compliant communications protocols (e.g., http or ftp), perhaps with additional
918 authentication/permission layers.

919

10.4 Overall Registry/ Repository Functional Requirements

921

922 10.4.1 Both centralized and decentralized repository models SHALL be acceptable within
923 the ebXML infrastructure. The ebXML registry and repository architecture SHALL also
924 support distribution.

925

926 10.4.2 At minimum, an ebXML compliant Registry/Repository System MUST be able to
927 accomplish the following Use Case Scenarios. The Registry/Repository Specification
928 MAY also append additional Use Cases to this list, however, this list is deemed essential
929 and therefore MUST be present in all Registry/Repository Systems. (see Appendix A –
930 “Use Case Elaborations” for more details)

- 931
- 932 **UC100** A Company Actor can Register as an SO (Submitting Organization [participant])
- 933 **UC102** A Trading Partner Actor publishes a Trading Partner Profile to a Registry
- 934 **UC103** A Registry updates its' index to reflect changes in a Trading Partner Actors' TPP document
- 935 **UC104** A Human Actor queries a Registry for a Specific Trading Partners' TPP
- 936 **UC105** A Human Actor can query a Registry for a non specific Trading Partner Actors' TPP
- 937 **UC106** An Application Actor can query a Registry for a Trading Partner Actors' TPP
- 938 **UC110** A Trading Partner Actor publishes a Business Process to a Registry
- 939 **UC115** A Human Actor can review a copy of a Repository Item in a GUI
- 940 **UC117** An Application Actor can query the Repository for a Core Component
- 941 **UC118** A Human Actor can Query the Repository for a Repository Item on criteria other than GUID
- 942 **UC120** A Human Actor can submit a concatenation or modification to a Core Component to a RA
- 943 (Repository Authority)
- 944 **UC124** A Registry/Repository Authority can Modify a Core Component
- 945 **UC125** The Repository can accept the Queries in an ebXML compliant message
- 946 **UC126** A Registry Actor communicates with another Registry Actor
- 947 **UC127** The Registry/Repository communicates an exception error to the RA.

948
949

950 10.4.3 A registry shall also be able to track and recognize *Data Stewards* and *Submitting*
951 *Organizations* of repository items and the repository items themselves.

952

953 10.4.4 A registry shall incorporate a mechanism for querying a *repository* or an index of
954 a repository via an API (Application Programming Interface)

955

956

957 **10.5 Requirements for Registry/Repository Authorities**

958

959 **10.5.1 Introduction**

960

961 A Registry/Repository Authority is a special Actor who has authority over a Registry
962 and/or Repository domain. The RA has special responsibilities and duties to ensure the
963 uninterrupted service provided by a Registry and/or Repository. The requirements for
964 becoming an RA SHALL be specified in the ebXML Registry and Repository
965 Specifications.

966

967 **10.5.2 Certification**

968

969 There are no plans to implement certification procedures for Registry and/or Repository
970 Authorities at this time. This MAY be mandated by the Registry/Repository Project
971 Team.

972 **10.5.3 RA Requirements**

973

974 The Registry and Repository Specification MUST address requirements for the
975 Repository Authority to clearly publish its' policies.

976

977 The Registry and Repository Specification MUST address requirements for an RA to
978 review all submissions to avoid duplication of Repository Items. This specifically refers
979 to submissions of Business Processes, DTD's/Schemas, Common Business Objects and
980 extensions to the Core Component library as submitted data elements. The submissions
981 will be formatted as per the ebXML Registry/Repository Specification.

982

983 The Registry and Repository Specification MUST address requirements for an RAs'
984 abilities to demonstrate Security knowledge.

985

986 The Registry and Repository Specification MUST address requirements for a Registry to
987 have a minimum fail safe back up system.

988

989 **10.6 Repository Items**

990

991 All repository items MUST have a *Data Steward, Submitting Organization* and a
992 *Repository Authority* associated with them. The *Data Steward* SHALL be the custodian
993 of the repository item. The Submitting organization SHALL be the intellectual property
994 owner of the Repository Item.

995

996 A submission to a registry/repository involves three stages of classification. The
997 repository items can be classified as:

998

999 Development view:

- 1000 ▪ *Submissions* - newly submitted items that need to be examined. Items at this
1001 stage MAY be, but are not limited to, UML diagrams.
- 1002
- 1003 ▪ *Work in progress* - a stage whereby the RA MUST examine and MAY
1004 suggest further refining of the repository items.
- 1005

1006

1007 Run time view:

1008

- 1008 ▪ *Standardized Items* - Repository items MUST now be expressed in *Valid*
1009 XML syntax and are available to parties other than the RA and the SO.
- 1010

1010

1011 **10.7 Registry and Repository – Distributed vs. Centralized**

1012

1013

1014

1015 10.8.1 Introduction

1016

1017 Several groups have advocated a symbiotic Repository/Registry relationship. The
1018 Repository functions much like a database by containing the items and XML documents
1019 which provide the necessary information that promotes interoperability. The Registry acts
1020 like an index and gateway to a Repository. It is generally accepted that a Registry MUST
1021 be able to index (Subscribe) to more than one Repository. The current views can be
1022 summarized as a centralized approach to building a Repository. This model is inefficient
1023 for the reason that it mandates high overhead for submitting, querying and retrieving
1024 items. For example some repository items, such as UML diagrams of works in Progress,
1025 may be several hundred kilobytes. Furthermore, the Centralized “Hub and Spoke” model
1026 does not scale well based on the anticipated overhead of a centralized Repository system.

1027

1028 The current views of Registry and Repository functionality also violate one of the key
1029 rules for architecting ebXML – “There SHALL NOT be any one single point of failure
1030 within the ebXML infrastructure”. A centralized Repository model certainly is a likely
1031 target for “denial of service” type attacks and can therefore be deemed a likely single
1032 point of failure candidate that could bring the entire infrastructure to a standstill.

1033

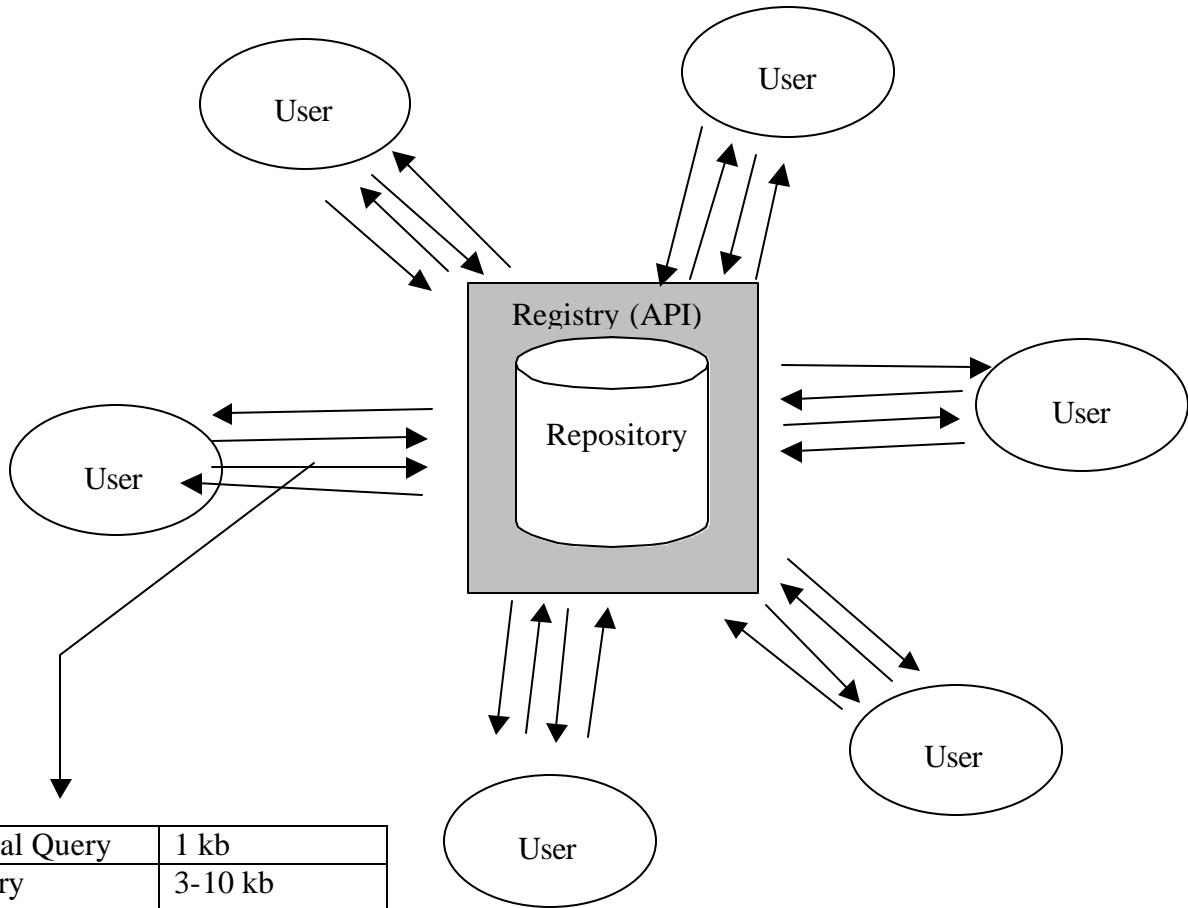
1034

1035 There SHALL be multiple repositories worldwide. This creates a problem because the
1036 Registries that subscribe to these Repositories MUST be able to communicate with one
1037 another to synchronize their contents. This problem has not been generally addressed yet
1038 due to a general lack of deployment and lack of run time knowledge.

1039

1039
 1040
 1041
 1042

10.8.2 Current Centralized Repository Approach (applicable to Core Component and DTD/Schema Registry/Repository Registry/Repository models)



1	Initial Query	1 kb
2	Query Response	3-10 kb
3	“Get” Request	1 kb
4	Response	50 kb+

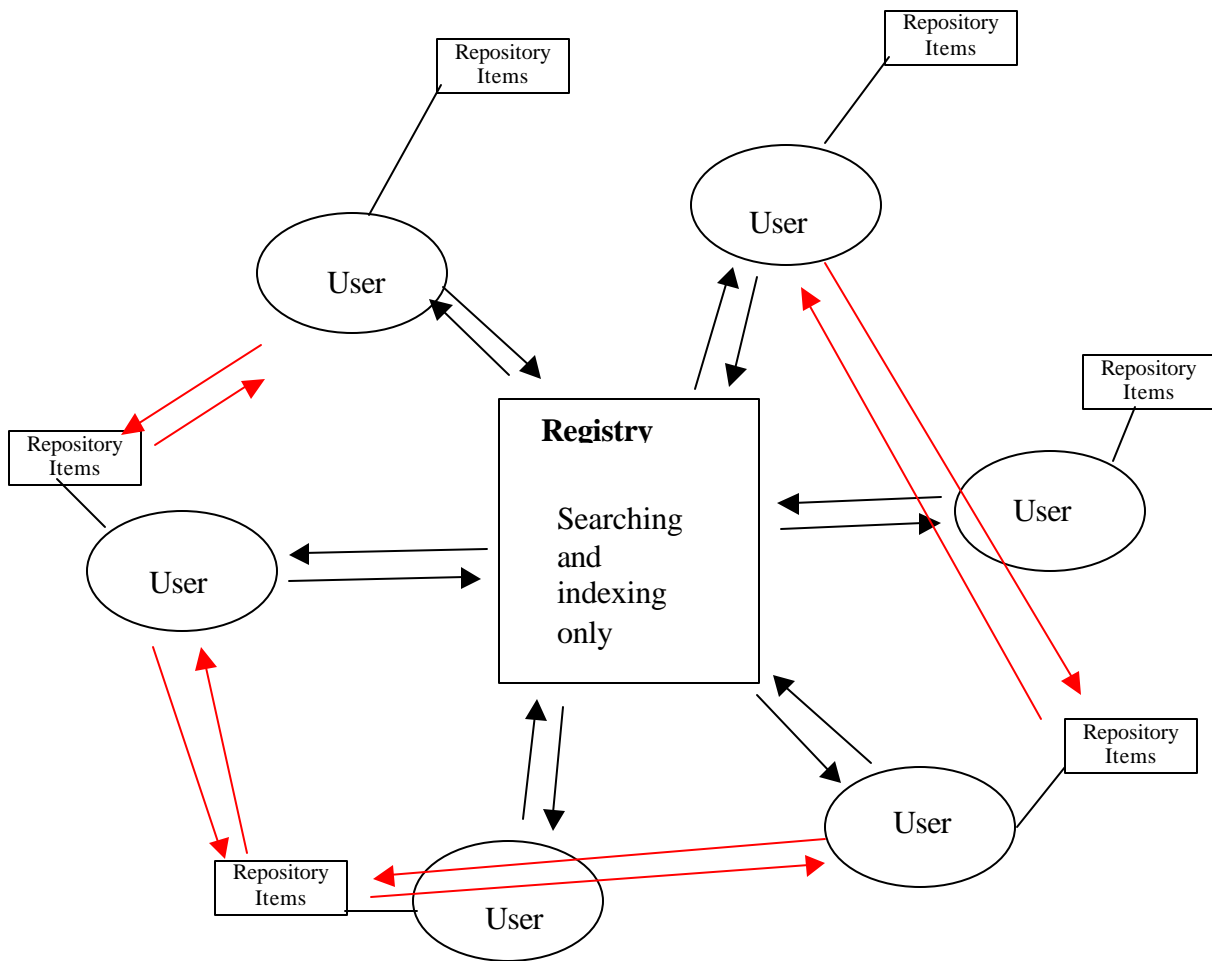
Notes: The Repository Items are centralized and kept in one central location. This model has several disadvantages as precipitates a high amount of I/O bandwidth overhead as well as internal process overhead. This model mimics the database centric viewpoint of placing all items into a single container. A Registry transaction is also required for each submission, modification request and query/authentication.

The biggest foreseeable burden is the “get” type requests to retrieve a Repository Items. This chilles heel” of the system and represents a potential single point of failure which is deemed unacceptable. Backup system could be engaged to mitigate that likelihood and employ failsafe, rollover type back up which must be synchronized. This represents a very high cost solution yet – may be favorable in certain instances.

In some cases, this model may be favorable for security and control reasons. For a Core Component Registry/Repository, this MAY be the desired model/

1043 **10.8.3 A New Model for a Distributed Repository (applicable to Trading Partner**
 1044 **Profile, Business Process document and DTD/Schema Registry/Repository models)**

1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053
 1054

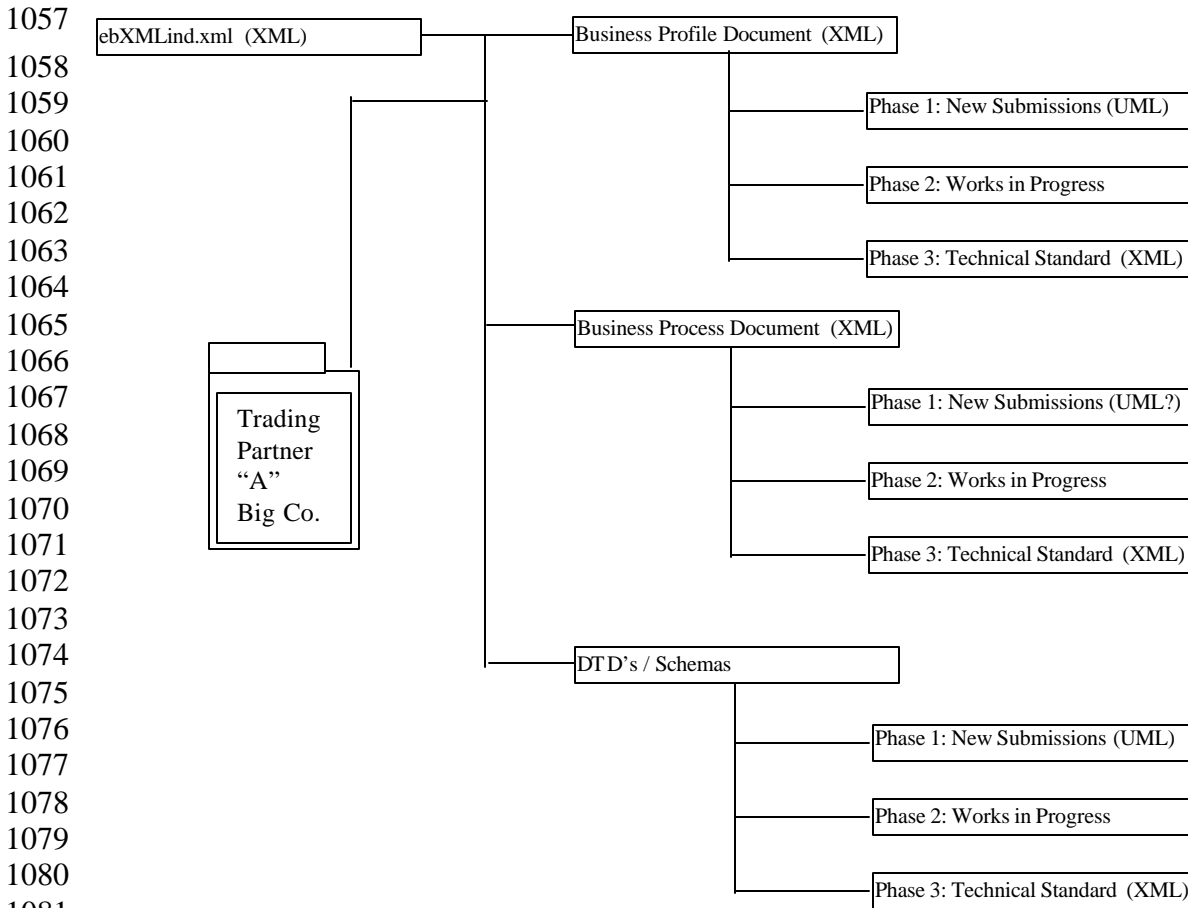


The Repository Items are now de-centralized and kept in web accessible directories by each SO. This model has several advantages as it allows each SO easy access to it's submissions and works in progress. An Agent acts for the Registry and visits each Item bound for submission. It is then "suggested" to the RA

The biggest benefit is the offloading of "get" type requests (shown in lighter colors) to the perimeter entities labeled "User" (Trading Partner Actors). This makes each SO responsible for their own bandwidth use and security, much the same as they are doing now for their websites and legacy systems.

SME's may not be using this model. SME integration will be discussed later.

1054 The previous examples of document instances MAY be stored somewhere and indexed
 1055 by the Registry. A simple system of filing in a directory structure, accessible via the
 1056 web, has been chosen for simplicity.



1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098

Figure X.X – Decentralized Repository Directory Structure – A special Actor called the Server Administrator Actor (extended from Trading Partner Actor) MUST set the access permissions to provide a Registry Agent access to the directory trees which contain the documents that the Trading Partner Actor wishes to submit to the Registry. A Registry Agent shall have access to those files for purposes of indexing them. The Registry Agent shall utilize a special valid xml file named ebXMLind.xml (see Appendix B - “Sample ebXMLind.xml file”) which a Registry Agent MAY use as a method to determine which items are to be included in the Registry’s index. The Trading Partners kit will also contain explanations of how to publish and change a special file called “ebXMLind.xml”. This file SHALL publish meta data related to new additions or changes to existing items being indexed by any ebXML compliant registry. The Registry MUST force its’ agents to parse this file before indexing any of the documents described above in section 2(a-c).

A repository item MAY go through several transitions as it moves from being a new submission to a Technical Standard.

New Submissions:

1099 The New Submission phase is centric to capturing the original concept of the Item and
1100 submitting it to the Repository Authority. The RA SHALL review all new submissions
1101 to identify redundant submissions. A New Submission, particularly of a Business
1102 Process, MAY be in the form of a UML diagram (ie – binary file) and therefore if it is
1103 bound for the Registry index, it must provide corresponding metadata to the Registry in
1104 valid XML syntax.

1105

1106 **Works in Progress**

1107

1108 A Work in Progress MAY be in the form of a UML diagram (ie – binary file) and
1109 therefore if it is bound for the Registry index, it must provide corresponding metadata to
1110 the Registry in valid XML syntax. It MAY be a Valid XML file at this stage.

1111

1112 **Technical Standard**

1113

1114 The third phase of a Repository Item life cycle MUST be expressed in Valid XML
1115 syntax. The OMG (Object Management Group) XMI (Xml Metadata Interchange)
1116 methodology MAY be used for transition of a Repository Item from a UML diagram to
1117 Valid XML syntax. The Requirements for the transformation methodology are as
1118 follows:

1119

- 1120 i) The methodology used MUST yield consistent results
- 1121 ii) The methodology MUST use a stable, open and published
- 1122 standard for such transformations

1123

1124

1124
1125

1126 **10.9 Registry and Repository Security Requirements**

1127

- 1128 a. Appropriate security protocols SHALL be deployed to offer authentication, protection
1129 and non-repudiation functionality to a repository and registry. A suitable security
1130 protocol SHALL be employed to ensure that system corruption or breaches are
1131 avoided. Specifically, the O/S shall be safeguarded against all reasonably foreseeable
1132 threats. All such Security Requirements SHALL be specified in the ebXML
1133 Registry/Repository Specifications.
1134
- 1135 b. At the time an Repository Item is first indexed in a Registry, or updated, the
1136 authentication SHALL identify the *Submitting Organization* and ascertain whether
1137 or not that entity has the authority to perform the requested actions.
1138
- 1139 c. The registry and repository shall track the necessary information provided by the
1140 transport layer to archive information for purposes of non-repudiation.
1141
- 1142 d. The *Repository Authority (RA)* shall make known what security procedures are being
1143 followed by clearly publishing its' policies and security procedures.
1144
- 1145 e. The *RA* is responsible for ensuring the security protocol is fully operational and
1146 SHALL take all reasonable steps to ensure a fully secure service.
1147
1148

1149 **10.10 Addressing Registry/Repository Items**

1150

- 1151 a. Repository Items SHALL be addressable as a URI in order to allow HTTP as the
1152 default transportation protocol.
1153
- 1154 a. b. Each Repository Item SHALL be identified by a Globally Unique Identifier.
1155 This GUID SHALL be built using a combination of URN and a CDATA suffix that
1156 SHALL not exceed 8 bytes in length. (see section 9.0 – “**Core Components**” for an
1157 example)

1158

1159 **10.11 Querying Registries**

1160

- 1161 a. A Registry Query mechanism SHALL be employed to query for Repository Items by
1162 either an Application Actor or a Human Actor (see Appendix A -“**Elaborated Use**
1163 **Cases**”).
- 1164
- 1165 b. An ebXML search mechanism should provide a web search interface for humans to
1166 manually locate data (particularly important for SMEs) and an API for applications to
1167 rapidly access stored data.
- 1168
- 1169 c. A caching mechanism is recommended to increase the efficiency of a query
1170 mechanism.
- 1171

1172 **10.12 ebXML Decentralized Repository Query Requirements**

1173

- 1174 a. In instances of Registries for Decentralized Repositories, a Context Sensitive XML
1175 search mechanism MUST be employed to provide a searchable index for Trading
1176 Partner Profile, Business Process and DTD/Schema metadata documents. This section
1177 refers explicitly to XML documents that are not part of a Centralized ebXML
1178 repository.
- 1179
- 1180 b. A mechanism MUST be present to recognize updates to one of the Repository Items
1181 mentioned in (a) above and re-index the item(s).
- 1182
- 1183
- 1184
- 1185
- 1186
- 1187

1187

11.0 Messaging and Transport Requirements

1188

1189

11.1 Introduction

1190

1191 This section specifically deals with the function requirements for Messaging Services
1192 components of the ebXML infrastructure. The Messaging Services Specifications are
1193 available at <http://www.ebxml.org/>.

1194

1195

11.2 Messaging Services Description

1196

1196 The Messaging Service comprises three parts, an abstract interface, functions provided by
1197 the Messaging Layer, and mapping to underlying transport(s).

1198

1199 The Message Service provides ebXML with an abstract interface whose functions, at an
1200 abstract level, are defined in section 5.4.2. The concrete realization of this abstract
1201 interface are yet TBD, but it is anticipated they will be defined in a language independent
1202 manner.

1203

1204 The Messaging Service architecture in 5.4.3 presents a conceptual model of the functions
1205 of the Message Service Layer, see figure 4.2. The Messaging service provides reliable
1206 and secure delivery of ebXML messages over various underlying transport protocol(s). It
1207 supports simplex (one-way) and request/response (either synchronous or asynchronous)
1208 message exchange. It also provides security, i.e., authentication, access-control, integrity,
1209 non-repudiation, and privacy, and enforces the rules of the TPA (see section zzz for
1210 TPA).

1211

1212 The Messaging Service Layer also maps ebXML onto the underlying transport
1213 protocol(s). The Messaging Service Layer may be mapped onto any connection oriented
1214 (CO) transport protocol which is capable of supporting MIME (which is required by
1215 ebXML packaging, see 5.4.4). The mapping to the underlying transport protocol(s) is not
1216 specified by ebXML (implementors will provide a mapping to the transport protocols
1217 they choose to support).

1218

1219

Notes:

1220

1. While ebXML is oriented toward the Internet, there is no requirement that the
1221 transport protocol be an internet protocol – ebXML can be used with other
1222 networks in addition to the Internet.

1223

2. While not giving preference to any particular transport, some which have been
1224 mentioned in the course of TRP work are HTTP and SMTP, particularly for the
1225 support of SMEs, and CORBA and DCOM for object oriented operation.

1226

1227 The relation of the abstract interface, the Messaging Service, and the CO transport
 1228 service is shown in figure 4.0 below:

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

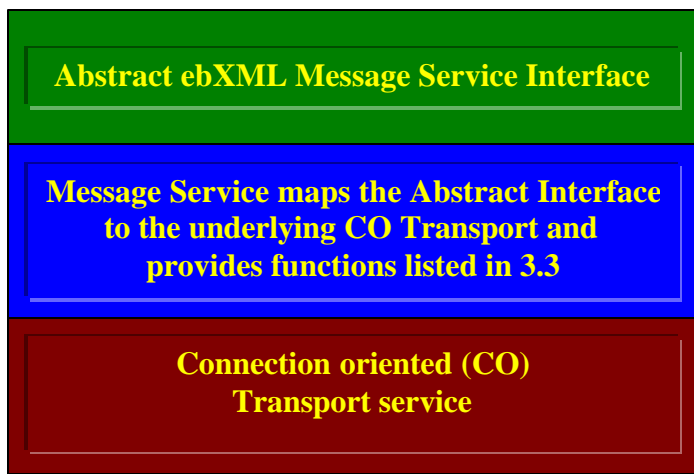


Figure 4.0 – [NOTE: add description for figure]

11.3 Abstract Messaging Services Interface

1256
 1257
 1258 The Message Service Layer abstract interface provides the following functions for the
 1259 exchange of ebXML messages:

- 1260
- 1261 ▪ Send – send an ebXML message – values for the parameters are derived from the
 1262 ebXML header, see X.X below.
- 1263 ▪ Receive – indicates willingness to receive an ebXML message
- 1264 ▪ Notify – provides notification of unexpected events, largely as a result of error
 1265 handling; these unexpected notifications are for events for which a Receive has
 1266 not been posted
- 1267 ▪ Inquiry – provides a method of querying the status of the particular Message
 1268 interchange
- 1269
- 1270

1271 **11.4 Messaging Layer Functions**

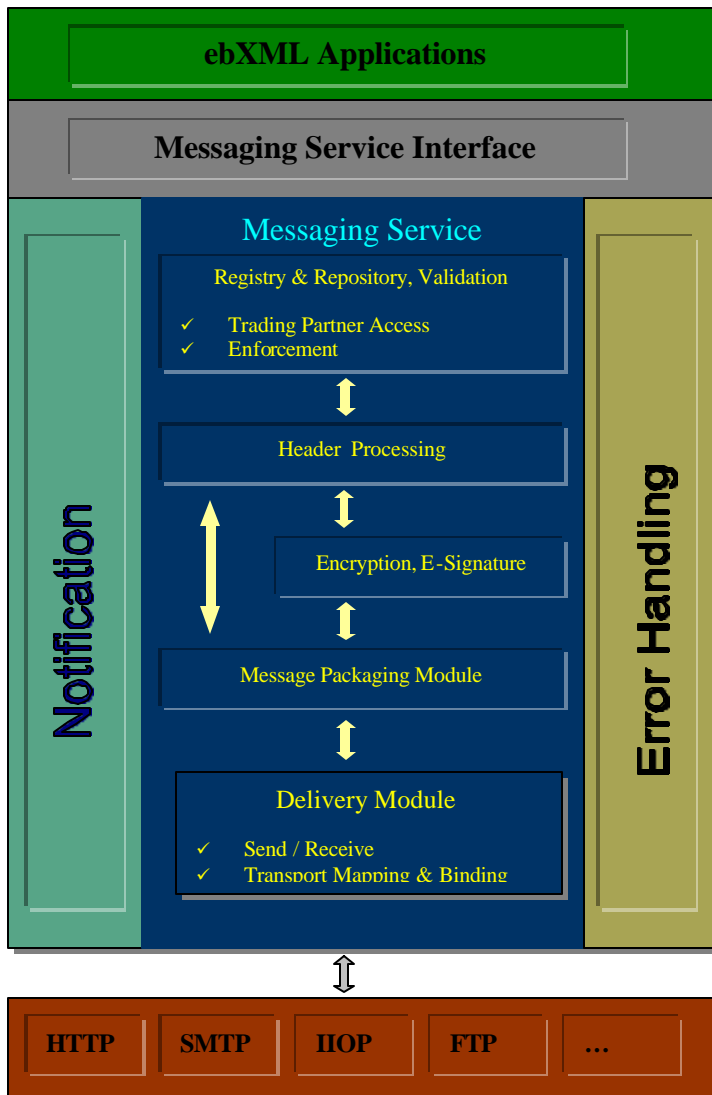
1272

1273 The Messaging Layer functions provide all the services needed to manage the entire
1274 lifecycle of ebXML messages. The list of functions provided by this layer includes:

- 1275 • The ability to construct and validate proper ebXML messages
- 1276 • The ability to acquire information about sending and receiving parties
- 1277 • Enforce the "rules of engagement" as defined by two parties in a TPA (including
1278 security, business process functions related to message delivery)
 - 1279 ○ It is assumed that parties wishing to engage in a business relationship will
1280 establish a set of "ground rules". These ground rules define acceptable
1281 behavior each party agrees to abide by. The definition of these ground
1282 rules can take many forms including formal Trading Partner Agreements,
1283 Interactive agreements established at the time a business transaction
1284 occurs (e.g buying a book online), or other forms of agreement. There are
1285 messaging services layer functions that enforce these "ground rules". For
1286 example two parties may agree to use encryption and digital signatures to
1287 protect business data. The messaging layer contains functions to ensure
1288 that all business data both in and out is encrypted and digitally signed.
1289 Any violation of the ground rules should result in an error condition,
1290 which is reported using the appropriate means.
- 1291 • Communicate messages between parties
 - 1292 ○ Parties will want the ability to choose the characteristics and behavior of
1293 message delivery depending on the business process and type of
1294 information being exchanged. The messaging services layer must support
1295 the following delivery options/behavior:
 - 1296 ▪ Unreliable delivery
 - 1297 ▪ Reliable delivery
 - 1298 ▪ At most once delivery of data
 - 1299 ▪ Synchronous and Asynchronous messaging
 - 1300 ▪ Request/Response processing mode
 - 1301 ▪ Fire'n'forget processing mode
 - 1302 ○ Allow for "multipart involvement" during message delivery
- 1303 • Perform all security related functions including:
 - 1304 ○ Access Control
 - 1305 ○ Privacy
 - 1306 ○ Authentication
 - 1307 ○ Integrity
 - 1308 ○ Non-repudiation
- 1309 • Interface with internal systems including:
 - 1310 ○ Presentation of data received from a trading partner to internal systems
 - 1311 ○ Error notification
- 1312 • Administrative Services
 - 1313 ○ Error tracking and reporting
 - 1314 ○ Notification, both system-to-system and system-to-human

- 1315 ○ Track and Report the status of message exchanges for auditing and
- 1316 diagnostic purposes
- 1317 ○ Auditing and Logging of non-delivery related conditions (e.g. system
- 1318 errors)
- 1319 ○ Access to trading partner information
- 1320 ○ Status inquiry
- 1321 ● Transport bindings
 - 1322 ○ The messaging service will include functions to enable the delivery of
 - 1323 messages over various transport services (e.g. SMTP, FTP, HTTP, etc.)

1324
 1325 The following diagram depicts a logical arrangement of the functional modules that exist
 1326 within the ebXML architecture. These modules are arranged in a manner to indicate their
 1327 inter-relationships and dependencies. The diagram also attempts to convey the
 1328 architectures flexibility, reflecting the spectrum of services that must be supported in
 1329 order to meet the anticipated usage patterns of ebXML implementers.



1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1360
 1361
 1362

1363
1364
1365

Figure 4.1 –The Messaging Services Architecture

1366 **11.5 Structure and Packaging**

1367

1368 The ebxml message structure separates information utilized by the Messaging Service
1369 from business related information. The former is contained in the message header and
1370 the latter in the payload. This separation promotes efficiency as the Messaging service
1371 only needs to access the message header and the payload can be passed transparently
1372 (without being processed by the Messaging Service). It also enables encrypted and/or
1373 signed payloads to be exchanged and forwarded transparently with no processing
1374 overhead. The ebXML Message Header is a mandatory part of each ebXML Message;
1375 ebXML Message Payload is optional. ebXML messages may be embedded within the
1376 payload of other ebXML messages (e.g., to allow forwarding and multicast).

1377

1378

1379 **ebXML Message Header structure**

1380

- 1381 a) ebXML Header carries Message Service related data and is ebXML Message
1382 Payload agnostic.
- 1383 b) ebXML Message Header shall be expressed in XML syntax (currently XML version
1384 1.0).
- 1385 c) the declaration shall be in the form required by the current version of XML to specify
1386 encoding the attribute as part of the ebXML Message Header, e.g., `<?xml`
1387 `version="1.0" encoding="UTF-8"?>`.
- 1388 d) ebXML Message Header shall be compliant to the Versioning, Naming and
1389 Extensibility approach defined in sections X.Y ... of this document.
- 1390 e) ebXML Message Header contains a message type. (The message type relates only to
1391 the Message Service; it is not related to the business semantics in the payload, e.g.,
1392 business processing error or business acknowledgement).
- 1393 f) ebXML Message Header shall contain a manifest which identifies the payload
1394 contents. The manifest facilitates payload processing and provides error checking.
- 1395 g) ebXML Message Header shall contain a globally unique identifier.
- 1396 h) The routing data shall be conveyed in the ebXML Message Header to support
1397 delivery and routing of the ebXML Messages between the involved parties.
- 1398 i) The business intent shall be conveyed in the ebXML Message Header to support
1399 delivery and routing of the business data to the appropriate business application or
1400 process.
- 1401 j) ebXML Message Header shall carry the information that defines the Message
1402 Serviceparameters that are necessary for the proper interchange between the involved
1403 trading parties.

1404

1405 For full description of the structure of ebXML header refer to “ebXML Messaging
1406 Service Message Header Specification”.

1407

1408 **ebXML Message Packaging**

1409

1410 a) In order to carry both XML and non-XML payload content, ebXML messages shall
1411 be packaged using MIME, see specification ebXML xxx for further information.

1412

1413

1414

1414

1415

12.0 ebXML Conformance and Testing

1416

12.1 Overview

1418

1419

1420 Objectives of this clause are to:

1421 a) ensure a common understanding of conformance and what is required to claim
1422 conformance;

1423

1424 b) promote interoperability and open interchange of business processes and
1425 messages;

1426

1427

1428 c) promote uniformity in the development of conformance tests.

1429

1430 Conformance to ebXML is defined in terms of conformance to ebXML, conformance to
1431 each of the component specifications for ebXML, and conformance to this (Technical
1432 Architecture) specification of ebXML.

1433

1434 All ebXML specifications shall contain a conformance clause. The conformance clause
1435 specifies explicitly all the requirements that have to be satisfied to claim conformance to
1436 that specification. These requirements may be applied and grouped at varying levels
1437 within each specification

1438

12.2 Conformance requirements

1440

1441 Types of conformance requirements can be classified as:

1442

1443 a) mandatory requirements: these are to be observed in all cases;

1444

1445 b) conditional requirements: these are to be observed if certain conditions set out in
1446 the specification apply;

1447

1448 c) optional requirements: these can be selected to suit the implementation, provided
1449 that any requirement applicable to the option are observed.

1450

1451 Furthermore, conformance requirements in a specification can be stated:

1452

1453 a) positively: they state what shall be done;

1454

1455 b) negatively (prohibitions): they state what shall not be done.
1456

1457 **12.3 ebXML Conformance**

1458
1459 ebXML Conformance is defined as conformance to an ebXML system that is comprised
1460 of all the architectural components of the ebXML infrastructure and satisfies at least the
1461 minimum conformance requirements for each of the ebXML component specifications.
1462

1463 **12.4 ebXML Component Specification Conformance**

1464
1465 Each ebXML component specification

- 1466
- 1467 a) shall support all the required functionality defined within this (TA) specification.
 - 1468
 - 1469 b) shall address who must conform, (e.g., implementations, applications).
 - 1470
 - 1471 c) may define multiple levels of conformance. These multiple levels are hierarchical
1472 in that a higher level of conformance entails conformance at all lower levels.
 - 1473
 - 1474 d) may provide additional or enhanced features or functionality (i.e., extensions) not
1475 required by the specification as long as those extensions are not explicitly
1476 prohibited in the specification. These non-standard extensions shall not alter the
1477 specified functionality or behaviours defined in the specification.
 - 1478

1479 **12.5 TA Conformance**

1480
1481 In order to conform to this specification, the ebXML component specifications

- 1482
- 1483 a) shall conform to all functional requirements defined in this specification;
 - 1484
 - 1485 b) shall not specify any requirements that would contradict or cause non-
1486 conformance to ebXML or any of its components;
 - 1487
 - 1488 c) may contain a conformance clause that adds requirements that are more specific
1489 and limited in scope than the requirements in this specification.

1490
1491

1492 **12.6 General framework of Conformance Testing**

1493

1494 The objective of conformance testing is to establish whether an implementation conforms
1495 to the requirements established by the ebXML specifications. Conformance testing can
1496 help to ensure correct implementation and utilization of standards and thus, increased
1497 probability of portability and interoperability of ebXML products and solutions.

1498
1499 Implementations and applications should be tested to verify their syntactic and semantic
1500 conformance to the ebXML Specifications.

1501
1502 Publicly available test suites from vendor neutral organizations such as OASIS, NIST,
1503 and W3C should be used to verify the conformance of ebXML implementations and
1504 applications claiming conformance to ebXML, to ensure that they are compliant with the
1505 base W3C Recommendations. For example, implementations with embedded XML
1506 processors should use the OASIS/NIST XML and DOM test suites (www.nist.gov/xml/)
1507 to verify the conformance to these W3C Recommendations.

1508

1509 **12.7 References**

1510

1511 [BL73] Bell, D.E. & LaPadula, L.J., "Secure Computer Systems:
1512 Mathematical Foundations and Model", Technical Report M74-
1513 244, The MITRE Corporation, Bedford, MA, May 1973.

1514

1515 [Bra97] Bradner, S., "Key words for use in RFCs to Indicate
1516 Requirement Level", BCP 14, RFC 2119, March 1997.

1517 **Disclaimer**

1518 The views and specification expressed in this document are those of the authors and are
1519 not necessarily those of their employers. The authors and their employers specifically
1520 disclaim responsibility for any problems arising from correct or incorrect implementation
1521 or use of this design.

1522

1522 **Contact Information**

1523

1524 Team Leader

1525 Name:

1526 Company:

1527 Street:

1528 City:

1529 State:

1530 Zip/other:

1531 Nation:

1532 Phone:

1533 EMail:

1534

1535 Vice Team Lead

1536 Name

1537 company

1538 Street

1539 city, state, zip/other

1540 Nation

1541

1542 Phone:

1543 EMail:

1544

1545 Editor

1546 Name

1547 company

1548 Street

1549 city, state, zip/other

1550 Nation

1551

1552 Phone:

1553 EMail:

1554

1554

1555 Copyright Statement

1556 Copyright © ebXML 2000. All Rights Reserved.

1557

1558 This document and translations of it may be copied and furnished to others, and
1559 derivative works that comment on or otherwise explain it or assist in its implementation
1560 may be prepared, copied, published and distributed, in whole or in part, without
1561 restriction of any kind, provided that the above copyright notice and this paragraph are
1562 included on all such copies and derivative works. However, this document itself may not
1563 be modified in any way, such as by removing the copyright notice or references to the
1564 Internet Society or other Internet organizations, except as needed for the purpose of
1565 developing Internet standards in which case the procedures for copyrights defined in the
1566 Internet Standards process must be followed, or as required to translate it into languages
1567 other than English.

1568

1569 The limited permissions granted above are perpetual and will not be revoked by ebXML
1570 or its successors or assigns.

1571

1572 This document and the information contained herein is provided on an
1573 "AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR
1574 IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE
1575 USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1576 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1577 PARTICULAR PURPOSE.

1578

13 *EbXML Elaborated Use Case Scenarios*

- 1578
1579
1580
1581
1582 **UC100** A Company Actor can Register as an SO (Submitting Organization [participant])
1583 **UC101** A Trading Partner Actor creates a Trading Partner Profile
1584 **UC102** A Trading Partner Actor publishes a Trading Partner Profile to a Registry
1585 **UC103** A Registry updates its' index to reflect changes in a Trading Partner Actors' TPP document
1586 **UC104** A Human Actor queries a Registry for a Specific Trading Partners' TPP
1587 **UC105** A Human Actor can query a Registry for a non specific Trading Partner Actors' TPP
1588 **UC106** An Application Actor can query a Registry for a Trading Partner Actors' TPP
1589 **UC107** A Human Actor can read a Trading Partner Actors' TPP
1590 **UC108** A Application Actor can parse a Trading Partner Actors' TPP
1591 **UC109** A Trading Partner Actor creates a Business Process in XML syntax.
1592 **UC110** A Trading Partner Actor publishes a Business Process to a Registry
1593 **UC111** A Human Actor can identify a Business Process supported by a Trading Partner Actor
1594 **UC112** An Application Actor can identify a Business Process from its' GUID (Globally Unique Identifier)
1595 **UC113** An Application Actor can identify a DTD (Schema) based on its' GUID
1596 **UC114** An Application Actor can identify a Core Component based on its' GUID
1597 **UC115** A Human Actor can review a copy of a Repository Item in a GUI
1598 **UC117** An Application Actor can query the Repository for a Core Component
1599 **UC118** A Human Actor can Query the Repository for a Repository Item on criteria other than GUID
1600 **UC119** Trading Partner Actors ebXML Application resolves a semantic equivalency or a Core Component
1601 (or equivalent) referenced in a business message instance
1602 **UC120** A Human Actor can submit a concatenation or modification to a Core Component to a RA
1603 (Repository Authority)
1604 **UC121** A Company Actors ebXML Business Application Constructs and exchanges a series of messages
1605 to complete a business transaction with another Company Actor
1606 **UC122** A Trading Partner Actor creates a directory Structure and configures the ebXMLind.xml file.
1607 **UC123** A semantic Mapping Specialist Actor can review a Registry submission
1608 **UC124** A Registry/Repository Authority can Modify a Core Component
1609 **UC125** The Repository can accept the Queries in an ebXML compliant message
1610 **UC126** A Registry Actor communicates with another Registry Actor
1611 **UC127** The Registry/Repository communicates an exception error to the RA.
1612 **UC128** A Human Actor models a business process in UML.
1613

1613

1614

APPENDIX “A”

1615

Elaborated Use Case Descriptions

1616

1617

UC 100 Company Actor registers as a SO (Submitting Organization)

1618

1619

USE CASE #	UC100
Brief Description	A Company Actor register as an SO (Submitting Organization) with the RA (Registry/Repository Authority). The SO submits the application electronically to the RA. The RA receives the application and can set up an account for the SO with default permissions.
Preconditions	The SO can access the Registration screen.
Success End Condition	SOs account is created and receives user ID and password for account configuration modification in the future. The SO also receives information about how to publish content to the Registry.
Failed End Condition	Supplier’s account is not created
Primary, Secondary Actors	SO, RA
Trigger	SO hits a submit button or equivalent after filling in the appropriate form. A web form MAY be used for simplicity but other application formats may be deployed in the future
MAIN FLOW	
M1	SO accesses registration form

M2	SO fills in the form which requires the following information: <ul style="list-style-type: none"> • Business legal name • Address • Telephone • Contact person name • Contact person e-mail • URL (root web address) for publishing content to Registry. • Userid • Password • Settings (to be defined later) • Other information (TBA)
M3	SO hits the submit button or equivalent
M4	The information is transmitted to the RA
M5	The success of the form being accepted is posted back to the SO client acknowledging the acceptance of the form information.
M6	The RA sends the SO back a username and password
VARIATIONS (Sub flows)	
S1	A security protocol MAY be used in the future for conditions M3 to M6
S2	
ERROR HANDLING (Alternative Flows)	
E1	Missing or bad information results in no account creation and prompts supplier to update and re-submit form
E2	Database or OS errors from information filtering and verification
E3	The RA server times out or emits another HTTP error code indicating the attempt to receive the transmitted information failed

1620
1621

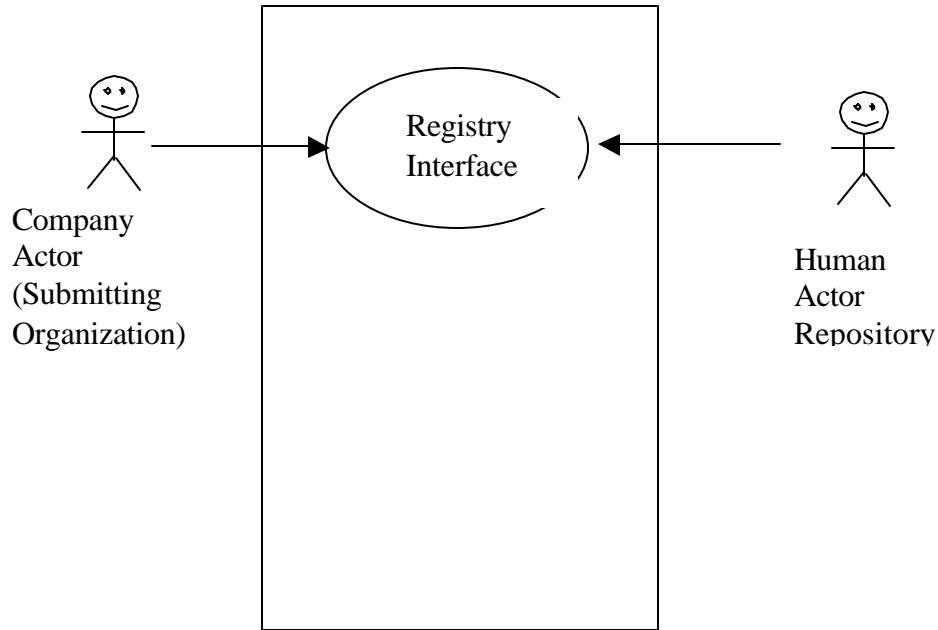
RELATED INFORMATION	
Priority:	Must have
Performance Target	GUI must be intuitive Process must be efficient (NOTE: security to be considered but added later)
Frequency	Unknown
Channels to actors	Interactive
OPEN ISSUES	The information required at registration time may change depending on the development of related technologies.
Due Date	
See also	User Login
Notes	<ul style="list-style-type: none"> • A CGI based methodology for capturing and transmitting information

	will likely be used in the first implementation. Later systems may contain windows apps etc. to submit the information. The system should be agnostic to the transportation of the information set.
--	---

1622

1623

1624 **UC100 SO Registration**



1625
1626
1627

UC101 A Trading Partner Actor creates a Trading Partner Profile

USE CASE #	UC101
Brief Description	An Trading Partner Actor creates a Trading Partner Profile to be indexed by the Registry
Preconditions	The Trading Partner is registered as a Submitting Organization with an ebXML certified and compliant Registry. The Trading Partner has the requirements for a set of minimal information which MUST be included in the Trading Partner Profile. Or, alternatively: The Trading Partner Actor has accessed a GUI form that they may enter the necessary information into which then automatically creates a Trading Partner Profile Document on their behalf.
Success End Condition	A Trading Partner Profile is created.
Failed End Condition	A TPP is not created
Primary, Secondary Human Actors	Company Actor
Primary Secondary Application actors	User Interface to Create TPP
Trigger	N/a
MAIN FLOW	
M1	Trading Partner Actor assesses the minimal requirements for a Trading Partner Profile Document which MAY be based on a DTD or Schema.
M2	The Trading Partner Actor places its' information in the correct format required by the ebXML TPP.
VARIATIONS (Sub flows)	
S1	The Trading Partner Actor accesses a GUI form and enters the necessary information to create a Trading Partner Profile document.
S2	

ERROR HANDLING (Alternative Flows)	
E1	The TPP must be validated against a DTD and rejected if not valid
E2	The information contained in the TPP must be semantically meaningful. All Business Processes must be identified by their GUID expressed as an attribute value which is a pointer to a Business Process referenced in an ebXML compliant Registry. If the reference is not present for each Business Process, the TPP must be rejected.

1628

RELATED INFORMATION	
Priority:	Must have
Performance Target	TBA
Frequency	Dependant on number of Repository users
OPEN ISSUES	
See also	
Notes	There may have to be some human process involved to ensure that the TPP contains all necessary information and references the correct Business Processes using the correct syntax. This represents a potentially costly and cumbersome process. A Software Vendor could make a simple set of business Processes available and easily facilitate adoption especially among SME's.

1629

1630

1630
1631
1632

UC102 A Trading Partner Actor publishes a Trading Partner Profile to a Registry

USE CASE #	UC102
Brief Description	An Trading Partner Actor publishes a Trading Partner Profile to a Registry. The Registry will essentially act as an XML Contextual Search Engine to allow other Trading Partner Actors to query the Trading Partner Actors Trading Partner Profile. This will result in a robust mechanism to facilitate dynamic interface discovery.
Preconditions	The Trading Partner is registered as a Submitting Organization with an ebXML certified and compliant Registry. The Trading Partner has successfully created a valid Trading Partner Profile document.
Success End Condition	A Trading Partner Profile is published to a Registry and accessible by other Trading Partner Actors.
Failed End Condition	A TPP is not published
Primary, Secondary Human Actors	Trading Partner Actor Registry Authority
Primary Secondary Application actors	User Interface to Publish a Trading Partner Profile
Trigger	N/a
MAIN FLOW	
M1	Trading Partner Actor presents its' SO credentials to a Registry Authority and receives confirmation they are recognized.
M2	Trading Partner Actor Reports to a Registry Interface the location of it's Trading Partner Profile.
M3	The Trading Partner Profile is indexed by the Registry's Agent.
VARIATIONS (Sub flows)	
S1	The Trading Partner Actor publishes its' TPP on its' own server and modifies the ebXMLind.xml document to point the Registry Agent to the location of the TPP document.
S2	

ERROR HANDLING (Alternative Flows)	
E1	All time outs and error handling for HTTP protocol errors must be handled
E2	An Agent for the Registry must ensure from time to time that the original TPP document is still in its' proper location.

1633

RELATED INFORMATION	
Priority:	Must have
Performance Target	TBA
Frequency	Dependant on number of Repository users
OPEN ISSUES	
See also	
Notes	There are two main models for creating a Repository of TPP documents. The models are outlined in the Registry and Repository Section 1.0 herein. The method favored for TPP indexing is the non centralized approach.

1634

1635

1635
1636
1637

UC103 A Registry Index is updated to reflect changes to a Trading Partner Profile

USE CASE #	UC103
Brief Description	From time to time, Trading Partner Actors may wish to update their Trading Partner Profiles as new Business Processes may be supported and old ones deleted or location of the document. This will be largely useful if someone queries the Registry in a non-unique way such as “Find all ebXML compliant companies who support Business Process XXX”. A Registry must facilitate the method for updating this information in a timely manner.
Preconditions	The Trading Partner has already published its’ TPP to a Registry.
Success End Condition	A Trading Partner Profile update is reflected in the Registry.
Failed End Condition	A TPP update is not reflected in the Registry
Primary, Secondary Human Actors	None
Primary Secondary Application actors	Registry Agent
Trigger	TBD – possibly a set schedule for revisiting indexed content OR the Trading Partner Actor may also trigger the event.
MAIN FLOW	
M1	The Registry sends an Agent to the location of the TPP document
M2	The Agent program retrieves the file and returns it to the Registry
M3	The Registry provides a mechanism which parses then indexes the TPP document and all associated meta data about it including its location
M4	The Registry then updates itself to reflect the changes
VARIATIONS (Sub flows)	
S1	The Trading Partner Actor publishes its’ TPP on its’ own server and modifies the ebXMLind.xml document to let the Registry Agent know it should reindex the TPP Document.
S2	
ERROR HANDLING (Alternative Flows)	
E1	All time outs and error handling for HTTP protocol errors must be handled

1638

E2	

1639

RELATED INFORMATION	
Priority:	Must have
Performance Target	TBA
Frequency	Dependant on number of Repository users
OPEN ISSUES	
See also	
Notes	

1639

1640 **UC104 Human Actor Queries a Registry for a Specific**
 1641 **Trading Partners' Trading Partner Agreement.**

1642

USE CASE #	UC104
Brief Description	A Human Actor can Query a Registry (Index) for a SPECIFIC Trading Partner Actors' Trading Partner Profile Document. The query must be contextual in nature.
Preconditions	Human Actor can access a query interface to a Registry. Human Actor knows at least one Unique Identifier to query upon for the Specific Trading Partner. (NOTE: This may likely be included as part of a hyper link or meta information to a catalog item that the second Actor may have viewed prior to conducting a query) A Trading Partner Profile has been successfully created by the Specific Trading Partner Actor and Indexed by a Registry pursuant to UC101 and UC102
Success End Condition	Human Actor receives enough information as a result of their query that they can retrieve the specific Trading Partners TPP document.
Failed End Condition	Human Actor does not receive a link to a Specific Trading Partners TPP document.
Primary, Secondary Human Actors	Human Actor (query initiator)
Primary Secondary Application actors	Registry Query daemon User Interface
Trigger	Human Actor activates a "Submit Query" button or equivalent
MAIN FLOW	
M1	Human Actor accesses a Search Interface
M2	Human Actor enters the GUID for the Specific Trading Partner
M2	Human Actor activates the "Submit" button or equivalent
M3	The search Criteria are sent to the Registry
M4	The query is resolved and transmitted to the Human Actor in the necessary format
M5	The Human Actor views the query results
VARIATIONS (Sub flows)	
S1	There are no results for the query. The human Actor is presented the appropriate message.

S2	
ERROR HANDLING (Alternative Flows)	
E1	Data process flow errors: time outs, no known host etc. as per HTTP or equivalent
E2	Data verification
E3	Two or more query results must be reported to RA

1643

RELATED INFORMATION	
Priority:	Must have
Performance Target	Under 5 seconds would be reasonable.
Frequency	Dependant on SO enrollment use case
OPEN ISSUES	
See also	
Notes	

1644

1645

1645
1646
1647

UC105 Human Actor Queries a Registry for a non Specific Trading Partners' Trading Partner Agreement.

USE CASE #	UC105
Brief Description	A Human Actor can Query a Registry (Index) for a non specific Trading Partner Actors' Trading Partner Profile Document. The query MAY be contextual in nature.
Preconditions	Human Actor can access a query interface to a Registry. Human Actor knows at least one search criteria to query upon.
Success End Condition	Human Actor receives enough information as a result of their query that they can retrieve one or more TPP documents.
Failed End Condition	Human Actor does not receive a link to a TPP document.
Primary, Secondary Human Actors	Human Actor (query initiator)
Primary Secondary Application actors	Registry Query daemon User Interface
Trigger	Human Actor activates a "Submit Query" button or equivalent
MAIN FLOW	
M1	Human Actor accesses a Search Interface
M2	Human Actor enters the search Criteria in the appropriate location in the GUI
M2	Human Actor activates the "Submit" button or equivalent
M3	The search Criteria are sent to the Registry
M4	The query is resolved and transmitted to the Human Actor in the necessary format
M5	The Human Actor views the query results
VARIATIONS (Sub flows)	
S1	There are no results for the query. The human Actor is presented the appropriate message.
S2	
ERROR HANDLING (Alternative Flows)	
E1	Data process flow errors: time outs, no known host etc. as per HTTP or equivalent
E2	Query character legality check

1648

RELATED INFORMATION	
Priority:	Must have
Performance Target	Under 5 seconds would be reasonable
Frequency	Dependant on SO enrollment use case
OPEN ISSUES	
See also	
Notes	This may be useful for locating partners within specific industry verticals or to interface with only partners who use certain business information sets in their Business Processes.

1649

1650

1651 ***UC106 An Application Actor can query a Registry to a***
 1652 ***Trading Partner Actors TPP document***

1653

USE CASE #	UC106
Brief Description	An Application Actor can query a Registry for a Trading Partner Actors' TPP document. This automated type of discovery mechanism must be able to search on both unique and non unique semantic values
Preconditions	The Application Actor has received instructions to locate TPP agreements based on a set of criteria.
Success End Condition	The Application Actor receives a query response which includes results for the query
Failed End Condition	The application Actor does not receive any results or gets and error code
Primary, Secondary Human Actors	None
Primary Secondary Application actors	Application Actor Registry Actor
Trigger	Application Actor submits a query to a Registry
MAIN FLOW	
M1	Application Actor submits the query with appropriate search criteria to the Registry

M2	The Registry resolves the query
M3	The Registry transmits the query results to the Application Actor
VARIATIONS (Sub flows)	
S1	
S2	
ERROR HANDLING (Alternative Flows)	
E1	All time outs and error handling for HTTP protocol errors must be handled
E2	

1654

RELATED INFORMATION	
Priority:	Must have
Performance Target	Within 5 seconds would be reasonable
Frequency	Dependant on number of Repository users
OPEN ISSUES	
See also	
Notes	

1655

1656 **UC107** ***A Human Actor can read a Trading Partner Actors'***
 1657 ***TPP***

1658

USE CASE #	UC107
Brief Description	A Human Actor reads a copy of a Trading Partner Actors' TPP. SME's will likely have to rely on human intervention to read and act upon this information.
Preconditions	The Human Actor has found the TPP
Success End Condition	Human actor is viewing a copy of a TPP in a format where they may be able to interact with it further (ie – clicking on links?)
Failed End Condition	Human Actor cannot read the TPP
Primary, Secondary Human Actors	Human Actor

Primary Secondary Application actors	Repository (web server) User Interface
Trigger	Human Actor requests a copy of an item. Eg. - Possibly by activating a hyperlink expressed as a result of a Registry Query
MAIN FLOW	
M1	Human actor issues request to read a TPP
M2	The TPP is loaded into an application that the Human Actor can read.
VARIATIONS (Sub flows)	
S1	The Request is run against a local cache first to save overhead of Repository requests.
S2	
ERROR HANDLING (Alternative Flows)	
E1	Data process flow errors: time outs, no known host etc. as per HTTP or equivalent
E2	No item is retrieved and other http type codes eg "404"
E3	Parser errors

1659

RELATED INFORMATION	
Priority:	Must have
Performance Target	TBA
Frequency	Dependant on number of Repository users
OPEN ISSUES	
See also	
Notes	

1660

1661

1662 ***UC108 An Application Actor can parse a Trading Partner***
 1663 ***Actors TPP document***

1664

USE CASE #	UC108
Brief Description	An Application Actor that has retrieved a copy of a Trading Partner Actors' TPP document must be able to parse it and subsequently use a set of handlers to act upon different bits of information contained within. A Specific example may be to recognize each Business Process or Business Interface supported by the Trading Partner Actor and compare it to an internal Cache to see if these are bilaterally understood.
Preconditions	The Application Actor has retrieved a copy of the Trading Partner Actors TPP document as a result of a query as described in UC104 / UC105 /

	UC108
Success End Condition	The Application Actor is able to fully parse the TPP XML document
Failed End Condition	The application Actor is not able to parse the TPP XML Document and subsequently cannot use any of the information contained therein.
Primary, Secondary Human Actors	None
Primary Secondary Application actors	Application Actor
Trigger	Application Actors Logic Daemon or equivalent interprets rules which direct it to parse a specific instance of a TPP Document
MAIN FLOW	
M1	Application Actor parses a TPP document
M2	The Application Actor invokes handlers based on its parsing instructions
VARIATIONS (Sub flows)	
S1	
S2	
ERROR HANDLING (Alternative Flows)	
E1	Standard XML 1.0 (and subsequent version) procedures (ie – non valid)
E2	

1665

RELATED INFORMATION	
Priority:	Must have
Performance Target	Fast
Frequency	Dependant on number of Repository users
OPEN ISSUES	
See also	
Notes	

1666

1667

1668 **UC109 A Trading Partner Actor creates a Business Process**
 1669 **in XML Syntax**

1670 [NOT DONE]

1671 **UC110 An RA can modify a Core Component Repository**
 1672 **Item**

1673
 1674

USE CASE #	UC110
Brief Description	The RA can append elements to a Core Component. Once approved, the modifications are reflected in the Core Component at run time.
Preconditions	The RA has had a Semantic Mapping Specialist Actor review the proposed changes. The RA has received a request to append an element to the Core Component
Success End Condition	A modification is made to the Core Component
Failed End Condition	No modification is made to the Repository Item
Primary, Secondary Human Actors	Repository Authority
Primary Secondary Application actors	Repository Interface
Trigger	RA modifies/appends one or more characters to a Core Component and publishes it.
MAIN FLOW	
M1	RA accesses the repository item via the GUI
M2	RA modifies information in the appropriate area of the GUI
M3	RA activates a “request change” button or equivalent
M4	The modified information is transmitted to the Repository
VARIATIONS (Sub flows)	
S1	
S2	
ERROR HANDLING (Alternative Flows)	
E1	Data process flow errors: time outs, no known host etc. as per HTTP or equivalent
E2	The SO is sent a message indicating S1

1675

RELATED INFORMATION	
Priority:	TO be determined (may be deemed non critical)
Performance Target	Non critical
Frequency	Unknown
OPEN ISSUES	
See also	
Notes	The rules for what would be an allowable situation for modification of a Repository Item need to be determined. Security considerations will need to be addressed.

1676

1677 ***UC111 A Human Actor can identify a Business Process***
 1678 ***supported by a Trading Partner Actor***

1679 [NOT DONE]

1680

1681 ***UC112 An Application Actor can identify a Business Process***
 1682 ***based on its' GUID***

1683 [NOT DONE]

1684

1685 ***UC113 An Application Actor can identify a DTD (Schema)***
 1686 ***based on its' GUID***

1687 [NOT DONE]

1688

1689 ***UC114 An Application Actor can identify a Core Component***
 1690 ***based on its' GUID***

1691 [NOT DONE]

1692

1693 ***UC115 A Human Actor can retrieve a copy of a Repository***
 1694 ***Item in a GUID***

1695

USE CASE #	UC115
Brief Description	A Human Actor retrieves a copy of a Repository Item as the result of a request for that item.
Preconditions	Successful execution of a query as described in UC104 AND/OR UC105 which provides information for retrieving the item

Success End Condition	Human actor is viewing a copy of a Repository Item
Failed End Condition	Human Actor cannot retrieve Repository Item
Primary, Secondary Human Actors	Human Actor
Primary Secondary Application actors	Repository (web server) User Interface
Trigger	Human Actor requests a copy of an item. Eg. - Possibly by activating a hyperlink expressed as a result of a Registry Query
MAIN FLOW	
M1	Human actor issues request for a Repository Item
M2	Request is transmitted to the repository
M3	The repository transmits the copy of the repository item back to the Human Actor
VARIATIONS (Sub flows)	
S1	The Request is run against a local cache first to save overhead of Repository requests.
S2	
ERROR HANDLING (Alternative Flows)	
E1	Data process flow errors: time outs, no known host etc. as per HTTP or equivalent
E2	No item is retrieved and other http type codes eg "404"

1696

RELATED INFORMATION	
Priority:	Must have
Performance Target	TBA
Frequency	Dependant on number of Repository users
OPEN ISSUES	
See also	UC104 UC105
Notes	This may result in a "GET" type request being sent to either a centralized Repository (ie – database type structure) or a non centralized repository (ie – WEBDAV type system with a centralized Registry).

1697

1698 **UC116 An Application Actor can query a Repository for a**
 1699 **Core Component.**

1700 [NOT DONE]
 1701

1702 **UC117 Human Actor Queries a Registry on criteria other**
 1703 **than a GUID**
 1704

USE CASE #	117
Brief Description	A Human Actor can Query the Registry for an XML syntax Repository Item on a value other than it's GUID. The query must be contextual in nature.
Preconditions	Human Actor can access a query interface.
Success End Condition	Human Actor results of their query.
Failed End Condition	Query results do not get returned to the Human Actor
Primary, Secondary Human Actors	Human Actor (query initiator)
Primary Secondary Application actors	Registry Query daemon User Interface
Trigger	Human Actor activates a "Submit" button or equivalent
MAIN FLOW	
M1	Human Actor accesses a Search Interface. The Search Interface may be built using Schemantix which will build a form based on the Schema or DTD structure of the Repository Items
M2	Human Actor enters the search criteria
M2	Human Actor activates the "Submit" button or equivalent
M3	The search Criteria are sent to the Registry
M4	The query is resolved and transmitted to the Human Actor in the necessary format
M5	The Human Actor views the query results
VARIATIONS (Sub flows)	
S1	Unknown
S2	
ERROR HANDLING (Alternative Flows)	
E1	Data process flow errors: time outs, no known host etc. as per HTTP or equivalent
E2	Data verification

1705

E3	
-----------	--

RELATED INFORMATION	
Priority:	For phase one it is not deemed mission critical
Performance Target	TBA
Frequency	Dependant on SO enrollment use case
OPEN ISSUES	
See also	
Notes	Schema Express is suited to do this job

1706

1707

1708 ***UC118 An Application Actor can resolve a semantic***
 1709 ***equivalency of a Core Component (or equivalent) referenced in a***
 1710 ***business message instance.***

1711 [NOT DONE]

1712

1713 ***UC119 A Human Actor can submit a concatenation of a Core***
 1714 ***Component to the RA***

1715

USE CASE #	US119
Brief Description	A Human Actor can make a submission to append elements to a Core Component XML document (Repository Item). The purpose is to allow vocabulary owners the ability to suggest mapping elements be added to facilitate interoperability between disparate XML vocabularies.
Preconditions	Human actor has retrieved a copy of the Repository Item
Success End Condition	An element or elements are submitted to the Core Component RA to be appended to the repository item.
Failed End Condition	No element or elements are submitted to the RA
Primary, Secondary Human Actors	Human Actor, Repository Authority
Primary Secondary Application actors	Repository Interface
Trigger	Human Actor activates a “send” button or equivalent

MAIN FLOW	
M1	Human Actor accesses the Core Component via the GUI
M2	Human Actor appends information in the appropriate area of the GUI
M3	Human Actor activates a “save” button or equivalent
M4	The information is transmitted to the RA
M5	The Human Actor is notified that the RA received his submission
VARIATIONS (Sub flows)	
S1	Unknown
S2	
ERROR HANDLING (Alternative Flows)	
E1	Data process flow errors: time outs, no known host etc. as per HTTP or equivalent
E2	The Human Actor is sent a message indicating S1

1716

RELATED INFORMATION	
Priority:	TO be determined (may be deemed non critical)
Performance Target	Non critical
Frequency	Unknown
OPEN ISSUES	
See also	
Notes	The rules for what would be an allowable situation for appending an element to a Core Component Repository Item need to be determined.

1717 **UC121 – UC 128**

1718 [NOT DONE]

1719

1719

APPENDIX “B”

1720

Implementation Issues

1721

1722 ***B100 - Extending the ebXML Core Component Library***

1723

1724 A Trading Partner Actor may determine that the Core Component Library is not
 1725 sufficient to facilitate a robust set of business messages and therefore wish to add a new
 1726 **Data Element** to their business vocabulary. The Core Component Specification, in
 1727 conjunction with the Registry and Repository Specification, SHALL provide clear
 1728 guidelines for modeling, authoring and submitting a new Data Element to an ebXML
 1729 compliant Registry/Repository.

1730

1731 The new submissions **MUST** be examined by a certified Repository Authority (see
 1732 section 10.0 – “**Registry Repository Functional Requirements and Overview**”)

1733

1734

1735 ***B101 - Using the Core Components in DTDs at Run Time***

1736

1737 c. Core Component repository reference SHALL be present in a DTD of a business
 1738 message instance and **MUST** be present for each occurrence of an XML element
 1739 within that Business message instance.

1740

1741 d. The Core Component Repository Reference in a DTD SHALL be expressed and
 1742 linked to each XML element using a fixed Attribute value.

1743

1744 e. The reference is there to provide either a machine or human with semantic
 1745 information about what the character data in the corresponding XML Element
 1746 represents.

1747

1748

XML FILE:

1749

1750

```

<?xml version="1.0"? encoding="UTF-8">
<DOCTYPE TelephoneInfo SYSTEM TelephoneInfo.dtd>
<TelephoneInfo>
  <TelephoneAreaCode>604</TelephoneAreaCode>
</TelephoneInfo>
```

1751

1752

1753

1754

1755

1756

1757

1758

DTD FILE:

1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784

```
<?xml version="1.0"?>
<!DOCTYPE TelephoneInfo [
<!ELEMENT TelephoneInfo (TelephoneAreaCode)>
<!ELEMENT TelephoneAreaCode (#CDATA)>
<!ATTLIST TelephoneAreaCode type CDATA #FIXED
'ebXML:10001'
]>
```

- f. The preceding example allows the element to be associated with the ebXML Core Component with a GUID of “ebXML:10001”.
- g. The method for ebXML XML document instance references to specific data elements within a repository must be able to be accomplished by using either DTD’s or schemas (when schemas become eligible for inclusion within the ebXML Architecture). The exact syntax for using a schema to accomplish the functionality of the DTD file example (above) must adhere to the same functionality.
- h. For the purposes of forwards compatibility with schemas as described in (c) above, the term “DTD” shall be interchangeable with schema for the remainder of this section.
- i. An Actor, whether it be a Human Actor or an Application Actor, that reviews such an XML file, with a repository lookup definition in the accompanying DTD, MAY perform the repository lookup on the data element to find out what it is.

1785

1786 ***B300 - Context Issues***

1787

1788 ***Overview***

1789

1790 The following Context Issues affect Common Business Object only. They do not affect
1791 Core Components.

1792

1793 Expanding on the previous example (see figure below), a Trading Partner Actor sending a
1794 business message instance to another Trading Partner Actor who resides in a different
1795 country may (for this example) require that the Common Business Object be extended to
1796 include another Core Component which could be called “Country Code”. IN a similar
1797 example, if a Trading Partner Actor wishes to send a business message instance to
1798 another Trading Partner Actor in the same geographical proximity, the Core Components
1799 called “Country Code” and “Telephone Area Code” may not be required. These context

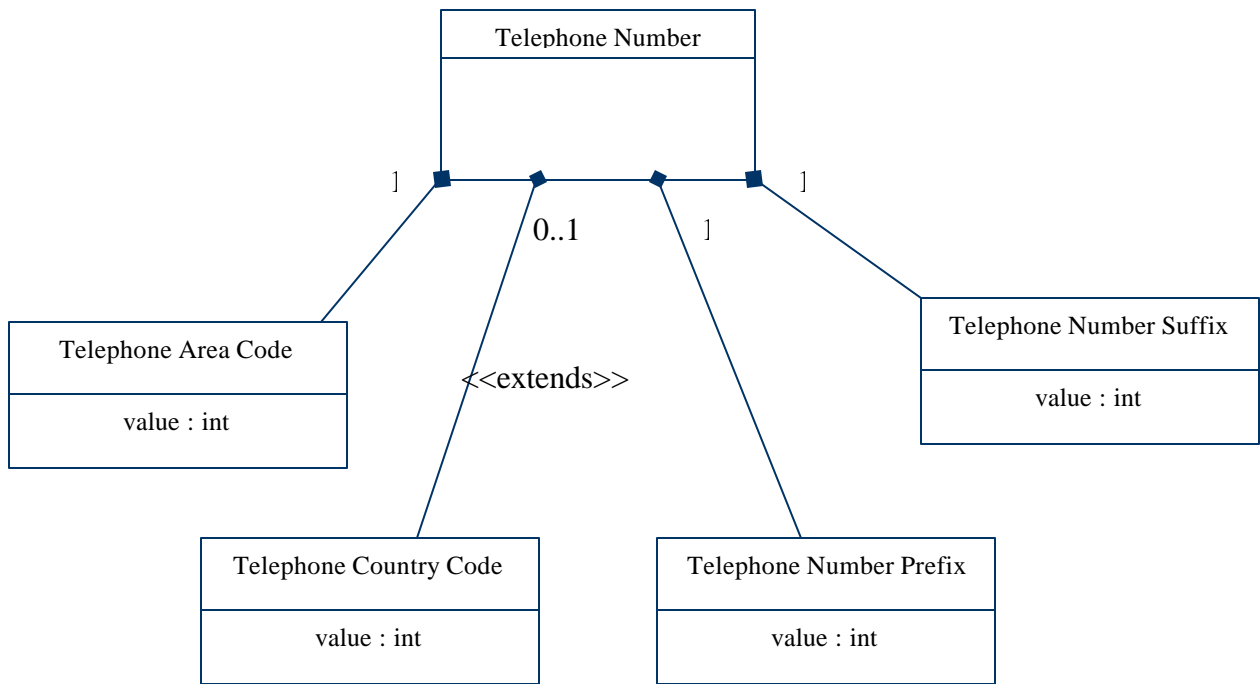
1800 requirement changes occurs at run time and an ebXML compliant Application MAY
1801 require a special Logic Engine component to analyze business data and apply rules based
1802 on Context.
1803

1804 ***B400 - Registry/Repository Interactions***

1805

- 1806 a. A registry SHALL have an interface to allow stewards to submit or update their
1807 repository items.
1808
- 1809 b. A classification scheme SHALL be employed to categorize repository items. These
1810 schemes SHALL be defined within the ebXML Registry and Repository technical
1811 specifications.
1812
- 1813 c. A registry SHALL provide an interface for applications (i.e., API) to query for
1814 repository items in a contextual manner. The syntax for this query SHALL be
1815 defined within the ebXML Registry and Repository Specification and/or the ebXML
1816 Transport, Routing and Messaging Technical Specification.
1817
- 1818 d. The API mentioned in (c) above, MAY also provide further functionality like
1819 allowing RPCs (Remote Procedure Calls) to modify repository items (with
1820 appropriate permissions). An example could be an XML syntax message delivered to
1821 the API that results in a query being performed against a Registry.
1822
- 1823 e. A registry SHALL provide an interface for humans (possibly CGI based) to manually
1824 query for repository objects. The query MUST be done in a contextual manner.
1825
- 1826 f. A Registry and Centralized Repository SHALL synchronize their contents with one
1827 another in a symbiotic “publish/subscribe” relationship. This is a synchronization of
1828 pointers from the Registry to the Repository, not a synchronization of content data.
1829
- 1830 g. Rules ‘e’ and ‘f’ (above) MUST be able to scale across multiple repositories.
1831
1832
1833

1833
 1834
 1835
 1836
 1837
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849



Example X.X – A Class view Diagram of a *Common Business Object* called “Telephone Number”. IN this example, the Common Business Object has been extended to include a new *Core Component* called “Telephone Country code”.

1849 **APPENDIX “C”**

1850 **Sample Files**

1851

1852 **Sample Trading Partner Profile XML File**

1853

1854 **[EDITORS NOTE: this will be replaced with document from TPP group. Consider**
 1855 **this a placeholder only to visualize the functionality of the document]**

1856

```

1857 <?xml version = "1.0" encoding="UTF-8">
1858 <!DOCTYPE Article PUBLIC "http://www.ebxml.com/dtds/tradingPartnerProfile.dtd">
1859 <tradingPartnerProfile xmlns:x = 'http://www.w3c.org/TR/xlink/working'
1860     xmlns:ebxml = 'http://www.ebxml.com/namespaces/reserved'>
1861     <!--this section is incomplete-->
1862     <Organization>
1863         <name>Trading Partner "A"</name>
1864         <subGroup>IT systems Group</subGroup>
1865         <contactInformation>
1866             <contactName>Duane Nickull</contactName>
1867             <address>1818 Cornwall Street</address>
1868             <city>Vancouver</city>
1869             <region type="province">British Columbia</region>
1870             <country>Canada</country>
1871             <postalCode>V6H-2W9</postalCode>
1872             <telephoneNumber>
1873                 <countryCode>01</countryCode>
1874                 <areaCode>604</areaCode>
1875                 <localNumber>717-1100</localNumber>
1876                 <extension>108</extension>
1877             </telephoneNumber>
1878         </contactInformation>
1879         <url>www.xmlglobal.com</url>
1880     </Organization>
1881     <companyProfile>
1882         <!--information in this section TBD later-->
1883         <typeOfCompany>manufacturer</typeOfCompany>
1884         <classification>some classification scheme</classification>
1885         <productsAvailable>shoes, foozeball table,
1886 skateboards</productsAvailable>
1887         <!--these element wil have to be agreed on using some standard-->
1888     </companyProfile>
1889     <securityInformation><!--security policy and requirements go here-->
1890 </securityInformation>
1891     <legalRequirements><!--legal requirements may go here-->
1892 </legalRequirements>
1893     <supportedProcesses>
1894         <process>
1895             <!--the GUID identifies a file in the BPM Registry-->
1896             <commonName GUID="ebxml:90001">Purchase Order</commonName>
1897             <description>Use this process for ordering our
1898 products</description>
1899         </process>
1900     </supportedProcesses>
1901 </tradingPartnerProfile>
    
```

```

1902         <commonName GUID="ebxml:90002">Order A Service</commonName>
1903         <description>Use this process to order a service.</description>
1904     </process>
1905     <process>
1906         <commonName GUID="ebxml:90003">Request a Quote</commonName>
1907         <description>Use this to request a quote from us</description>
1908     </process>
1909     <process>
1910         <commonName GUID="ebxml:90004">Shipping Advice</commonName>
1911         <description>Use this process for forwarding us shipping advice
1912 when you ship us stuff </description>
1913     </process>
1914     <process>
1915         <commonName GUID="ebxml:90005">Request for
1916 Information</commonName>
1917         <description>Use this process for requesting
1918 information</description>
1919     </process>
1920     <process>
1921         <commonName GUID="ebxml:90006">Request a Credit Note</commonName>
1922         <description>Use this for requesting a credit note</description>
1923     </process>
1924 </supportedProcesses>
1925 </tradingPartnerProfile>
1926
1927
1928
1929

```

1930 *Sample Business Process XML File*

1931

1932 **[EDITORS NOTE: this will be replaced with document from BPM group. Consider**
 1933 **this a placeholder only to visualize the functionality of the document]**

1934

1935

1936 <?xml version="1.0"? encoding="UTF-8">

1937 <!DOCTYPE Article PUBLIC "http://www.ebxml.com/dtds/simplePurchaseOrder.dtd">

1938 <businessProcess xmlns:x = 'http://www.w3c.org/TR/xlink/working'

1939 xmlns:ebxml = 'http://www.ebxml.com/namespaces/reserved'>

1940

1941 <process><!-- probably not complete yet-->

1942 <GUID>gxml4556</GUID>

1943 <processName>Simple Purchase Request</processName>

1944 <nameSpace>goxml</nameSpace>

1945 <!--the assigned URN can be used to globally and uniquely identify

1946 and retrieve this file from a repository-->

1947 <assignedURN>ebxml:14556</assignedURN>

1948 <partiesInvolved>2</partiesInvolved>

1949 <expiryDate mask="YYYYMMDD">20011231</expiryDate>

1950 <!--the concept of version may be deprecated. A different version

1951 constitutes

1952 a different component-->

1953 <version>1.0</version>

1954 </process>

1955 <SubmittingOrganization>

1956 <name>XML Global Technologies, Inc.</name>

1957 <subGroup>Technology</subGroup>

```

1958         <contactInformation>
1959         <!--it may be necessary to perform further semantic clarification in
1960 this section-->
1961         <contactName>Duane Nickull</contactName>
1962         <address>1818 Cornwall Street</address>
1963         <city>Vancouver</city>
1964         <region type="province">British Columbia</region>
1965         <country>Canada</country>
1966         <postalCode>V6H-2W9</postalCode>
1967         <telephoneNumber>
1968             <countryCode>01</countryCode>
1969             <areaCode>604</areaCode>
1970             <localNumber>717-1100</localNumber>
1971             <extension>108</extension>
1972         </telephoneNumber>
1973         </contactInformation>
1974         <url>www.ebxml.org</url>
1975     </SubmittingOrganization>
1976     <Roles><!--roles and identifiers go here-->
1977         <Role ID="Buyer">&entityReferencetoBuyer;
1978         </Role>
1979         <!--the Actor who publishes this process can easily identify themselves
1980             using entity references-->
1981         <Role ID="Vendor">&seller;
1982         </Role>
1983     </Roles>
1984     <step number="1">
1985         <Actor>Buyer</Actor>
1986     <!--the GUID identifies a file in the DTD Registry>
1987         <Object GUID="ebxml:50003">somePO.dtd</Object>
1988         <Action type="transmitsTo">&seller;</Action>
1989     </step>
1990     <step number="2">
1991         <Actor>&seller;</Actor>
1992         <Object GUID="ebxml:50003">someResponse.dtd</Object>
1993         <Action type="transmitsTo">Buyer</Action>
1994     </step>
1995     <description>blah blah</description>
1996 </businessProcess>
1997
1998
1999
2000

```

2001 *Example of a Core Component*

2002

2003 **[EDITORS NOTE: this will be replaced with document from Core Components**
 2004 **group. Consider this a placeholder only to visualize the functionality of the**
 2005 **document]**

```

2006 <?xml version="1.0" encoding="UTF-8"?>
2007 <!DOCTYPE CoreComponent SYSTEM "RepositoryItem.DTD" >
2008 <CoreComponent xmlns:x="http://www.w3c.org/TR/xlink/working"
2009             xmlns:ebxml="http://www.ebxml/repository/reserved">
2010
2011         <Organization OrgRole="RO">
2012             <orgID>ebXML</orgID>

```

```

2013     <orgFullName>Electronic Business XML Initiative</orgFullName>
2014     <Department>ebXML Liaison</Department>
2015     <OrgURL>www.ebXML.org</OrgURL>
2016     <AddrLine1>19191 Address Parkway</AddrLine1>
2017     <City>Cupertino</City>
2018     <CityParent ParentType="state">California</CityParent>
2019     <PostalCode>95014</PostalCode>
2020     <Country>USA</Country>
2021     <Contact ContactRole="Primary">
2022         <ContactName>Arofan Gregory</ContactName>
2023         <TelephoneNumber>
2024             <CountryCode>1</CountryCode>
2025             <AreaCode>408</AreaCode>
2026             <LocalNumber>517-3900</LocalNumber>
2027             <Extension/>
2028         </TelephoneNumber>
2029         <ContactEmail>arofan.gregory@commerceone.com</ContactEmail>
2030     </Contact>
2031
2032 </Organization>
2033 <Organization OrgRole="RA">
2034     <orgID>GoXML</orgID>
2035     <orgFullName>XML Global Technologies, Inc.</orgFullName>
2036     <Department>ebXML Registry</Department>
2037     <OrgURL>www.xmlglobal.com</OrgURL>
2038     <AddrLine1>1818 Cornwall Street, Suite #9</AddrLine1>
2039     <City>Vancouver</City>
2040     <CityParent ParentType="province">BC</CityParent>
2041     <PostalCode>V6J-1C7</PostalCode>
2042     <Country>Canada</Country>
2043     <Contact ContactRole="Primary">
2044         <ContactName>Duane Nickull</ContactName>
2045         <TelephoneNumber>
2046             <CountryCode>1</CountryCode>
2047             <AreaCode>604</AreaCode>
2048             <LocalNumber>717-1100</LocalNumber>
2049             <Extension>108</Extension>
2050         </TelephoneNumber>
2051         <ContactEmail>duane@xmlglobal.com</ContactEmail>
2052     </Contact>
2053 </Organization>
2054
2055
2056 <RegistryItemInstance
2057     ItemID="xc:10001"
2058     AssignedURN="urn:x-ebxml:org-xml:1"
2059     Version="2.0"
2060     ObjectLocation="http://repository.ebXML.org/ebxml_10001.xml"
2061     DefnSource="EBXML"
2062     PrimaryClass="defn"
2063     SubClass="xmlElem"
2064     MimeType="text/plain"
2065     RegStatus="reg"
2066     Stability="comp"
2067     ExpiryDate="20011231">
2068     <CommonName>TelephoneAreaCode</CommonName>
2069     <Description xml:lang="EN">The telephone Area Code core component is
2070 used to specify a dialing prefix needed for reaching the local telephone number
2071 from a region with a dissimilar area code.</Description>
2072
2073     <Description xml:lang="SP">La agencia disponible para tales como asignar

```

```

2074         identificadores numeros de pieza o codigos peligrosos de las
2075 mercancías.</Description>
2076         <ConstructedFrom>CDATA</ConstructedFrom>
2077
2078
2079         <EquivalentItems>
2080             <Item>xCBL:45561</Item>
2081             <!--This is for mapping semantic equivalencies-->
2082         </EquivalentItems>
2083
2084         <AssociatedItems>
2085             <!--reference by GUID later-->
2086             <CommonBusinessObjects>TelephoneNumber</CommonBusinessObjects>
2087         </AssociatedItems>
2088
2089
2090     </RegistryItemInstance>
2091
2092 </CoreComponent>
2093

```

2094 ***DTD – for above example***

```

2095
2096
2097 <!ENTITY % regStatusList
2098     "exp | reg | rep | sub | sup | wth" >
2099
2100 <!ENTITY % primaryClassList
2101     "defn | inst | pkg | other" >
2102
2103 <!ENTITY % relatedTypeList "CDATA" >
2104 <!--Fix later by specifying values-->
2105
2106 <!ENTITY % payStatusList "CDATA" >
2107 <!--Fix later by specifying values-->
2108
2109 <!ENTITY % defnSourceList
2110     " OASIS | IMS | IEEE_LOM | EBXML | UDDI |
2111 ProprietarySubmission " >
2112
2113 <!ENTITY % stabilityList
2114     "comp | dynm | stat">
2115
2116 <!ENTITY % subclassList
2117     " xmlDTD          | sgmlDTD | xmlSchema          | xdrSchema |
2118     soxSchema        | rdfSchema      | sgmlElement      |
2119     xmlElement       | sgmlAttrib   | xmlAttrib        |
2120     sgmlAttSet       | xmlAttSet   | sgmlAttVal       |
2121     xmlAttVal        | sgmlParm    | xmlParm          | charEntSet" >
2122
2123
2124 <!ELEMENT CoreComponent ( Organization+, RegistryItemInstance )>
2125
2126 <!ELEMENT Organization ( OrgId, OrgURN, OrgFullName, OrgCommonName?,
2127 Department?,

```

```

2128         OrgURL, ParentOrgURN?, ParentOrgId?, AddrLine1, AddrLine2?,
2129 AddrLine3?,
2130         City, PostalCode, Country, Email, Telephone, Fax? )>
2131 <!--ATTLIST Organization
2132         OrgRole      ( RO | RA | SO )  #REQUIRED >
2133 <!--ELEMENT OrgID (CDATA)>
2134 <!--ELEMENT OrgURN (CDATA)>
2135 <!--ELEMENT OrgFullName (CDATA)>
2136 <!--ELEMENT OrgCommonName (CDATA)>
2137 <!--ELEMENT Department (CDATA)>
2138 <!--ELEMENT OrgURL (CDATA)>
2139 <!--ELEMENT ParentOrgURN (CDATA)>
2140 <!--ELEMENT ParentOrgID (CDATA)>
2141 <!--ELEMENT AddrLine1 (CDATA)>
2142 <!--ELEMENT AddrLine2 (CDATA)>
2143 <!--ELEMENT AddrLine3 (CDATA)>
2144 <!--ELEMENT City (CDATA)>
2145 <!--ELEMENT CityParent (CDATA)>
2146 <!--ATTLIST CityParent ParentType ( State | Province | Region ) #IMPLIED
2147 >
2148 <!--ELEMENT PostalCode (CDATA)>
2149 <!--ELEMENT Country (CDATA)>
2150 <!--ELEMENT Email (CDATA)>
2151 <!--ELEMENT Telephone (CDATA)>
2152 <!--ELEMENT Fax (CDATA)>
2153
2154 <!--ELEMENT RegistryItemInstance
2155         (CommonName , Description+ , ConstructedFrom , EquivalentItems,
2156 AssociatedItems, Comment? )>
2157
2158 <!--ATTLIST RegItemInstance
2159         ItemId          ID                #REQUIRED
2160         AssignedURN     CDATA             #REQUIRED
2161         Version         CDATA             #IMPLIED
2162         ObjectLocation  CDATA             #IMPLIED
2163         DefnSource      (%defnSourceList;) #REQUIRED
2164         PrimaryClass    (%primaryClassList;) #REQUIRED
2165         SubClass        (%subClassList;)   #IMPLIED
2166         RelatedType    (%relatedTypeList;) #IMPLIED
2167         MimeType        CDATA             "text/xml"
2168         RegStatus       (%regStatusList;)  #REQUIRED
2169         StatusChg       CDATA             #IMPLIED
2170         Stability       (%stabilityList;)  #REQUIRED
2171         PayStatus       (%payStatusList;)  #IMPLIED
2172         ExpiryDate      CDATA             #REQUIRED >
2173
2174 <!--ELEMENT CommonName (CDATA )>
2175
2176 <!--ELEMENT Description (CDATA )>
2177 <!--ATTLIST Description lang (CDATA) #IMPLIED>
2178 <!--ELEMENT EquivalentItems (Item*)>
2179 <!--ELEMENT Item (CDATA)>
2180 <!--ATTLIST Item
2181         ItemParentVocabulary CDATA        #IMPLIED
2182         ItemParentVocabularyOrg CDATA      #IMPLIED >

```

```

2183 <!ELEMENT AssociatedItems (VocabularyCommonName,
2184 CommonBusinessObjects*)>
2185 <!ELEMENT VocabularyCommonName (CDATA)>
2186 <!ATTLIST VocabularyCommonName VocabularyID CDATA #REQUIRED >
2187 <!ELEMENT CommonBusinessObjects (CDATA)>
2188 <!ATTLIST CommonBusinessObjects
2189     GUID CDATA #REQUIRED >
2190 <!ELEMENT Comment (CDATA )>
2191
2192

```

2193 ***Example of a Common Business Object***

2194
2195

2196 **[EDITORS NOTE: this will be replaced with document from Core Components**
 2197 **group. Consider this a placeholder only to visualize the functionality of the**
 2198 **document]**
 2199

```

2200 <?xml version="1.0"? encoding="UTF-8">
2201 <!DOCTYPE Article PUBLIC "http://www.ebxml.com/dtds/commonObject.dtd">
2202 <commonObject xmlns:x = 'http://www.w3c.org/TR/xlink/working'
2203     xmlns:ebxml =
2204 'http://www.ebxml.com/CommonBusinessObjects/reserved' >
2205     <Item>
2206         <GUID>10201</GUID>
2207         <commonElementName extensible="YES">TelephoneNumber</commonElementName>
2208         <nameSpace>ebXML</nameSpace>
2209         <!--the assigned URN can be used to globally and uniquely identify
2210             and retrieve this file from a repository-->
2211         <assignedURN>ebxml:10201</assignedURN>
2212         <constructedFrom concatenator="#160;"><!--joinwith a space-->
2213         <!--this is an object composed of three Core Components in the repository
2214             in order of appearance in this branch.-->
2215         <item>
2216             <GUID>ebxml:10001</GUID>
2217             <commonElementName>TelephoneAreaCode</commonElementName>
2218             <x:link actuate="auto">www.myrepository.org/items/byGUID</x:link>
2219         </item>
2220         <item>
2221             <GUID>ebxml:10002</GUID>
2222             <commonElementName>TelephoneNumberPrefix</commonElementName>
2223             <x:link actuate="auto">www.myrepository.org/items/byGUID</x:link>
2224         </item>
2225         <item>
2226             <GUID>ebxml:10003</GUID>
2227             <commonElementName>TelephoneNumberSuffix</commonElementName>
2228             <x:link actuate="auto">www.myrepository.org/items/byGUID</x:link>
2229         </item>
2230     </constructedFrom>
2231     <expiryDate mask="YYYYMMDD">20011231</expiryDate>
2232     <version>1.0</version>
2233 </Item>
2234 <SubmittingOrganization>
2235     <name>ebXML Organization</name>
2236     <subGroup>Technical Architecture</subGroup>
2237     <contactInformation>
2238

```

```
2239         <!--it may be necessary to perform further semantic clarification in
2240 this section-->
2241         <contactName>Duane Nickull</contactName>
2242         <address>1818 Cornwall Street</address>
2243         <city>Vancouver</city>
2244         <region type="province">British Columbia</region>
2245         <country>Canada</country>
2246         <postalCode>V6H-2W9</postalCode>
2247         <telephoneNumber>
2248             <countryCode>01</countryCode>
2249             <areaCode>604</areaCode>
2250             <localNumber>717-1100</localNumber>
2251             <extension>108</extension>
2252         </telephoneNumber>
2253         </contactInformation>
2254         <url>www.ebxml.org</url>
2255     </SubmittingOrganization>
2256     <associatedItems>
2257         <!--these items are equivalents in different vocabularies-->
2258         <item>xcbl:5289</item>
2259         <item>visa:3995</item>
2260     </associatedItems>
2261     <description>This item is a common business object of the goxml library.
2262 The semantic
2263 meaning of the TelephoneNumber element is used as a container for the
2264 complete
2265 Telephone Number. It is comprised of core components listed within this
2266 document.
2267 It may be subject to being extended at run time due to contextual use.
2268     </description>
2269 </commonObject>
```