ebXML Technical Architecture Specification

ebXML Technical Architecture Team

17 October 2000

1.0 Status of this Document

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

This version:
>   ebXML_TA_v0.9.doc

*Latest version***:**
>   N/A

Previous version:
>   EbXML_TA_v0.8.72i

2.0 Scope

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

3.0 Normative References

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

W3C XML v1.0 specification

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

45   *4.0 ebXML Technical Architecture Participants*

46
47   We would like to recognize the following for their significant participation in the
48   development of this document.
49
50   Editors: Duane Nickull, XML Global Technologies
51   Brian Eisenberg, DataChannel
52
53   Participants: Colin Barham, TIE
54   Al Boseman
55   Dick Brooks, Group 8760
56   Cory Casanave, DataAccess Technologies
57   Robert Cunningham, Military Traffic Management Command, US Army
58   Christopher Ferris, Sun Microsystems
59   Anders Grangard, EDIFrance
60   Kris Ketels, SWIFT
61   Piming Kuo, Worldspan
62   Kyu-Chul Lee, Chungnam National University
63   Henry Lowe, OMG
64   Melanie McCarthy, General Motors
65   Klaus-Dieter Naujok, NextEra Interactive
66   Bruce Peat, eProcessSolutions
67   John Petit, KPMG Consulting
68   Mark Heller, MITRE
69   Scott Hinkelman, IBM
70   Karsten Riemer, Sun Microsystems
71   Lynne Rosenthal, NIST
72   Nikola Stojanovic, Columbine JDS Systems
73   Jeff Sutor, Sun Microsystems
74   David RR Webber, XML Global Technologies
75

5.0 Table of Contents

6.0 Introduction

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

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

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

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

180

7.0 ebXML Abstract Overview

182

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

188

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

200

8.0 ebXML Conceptual Overview

202

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

210
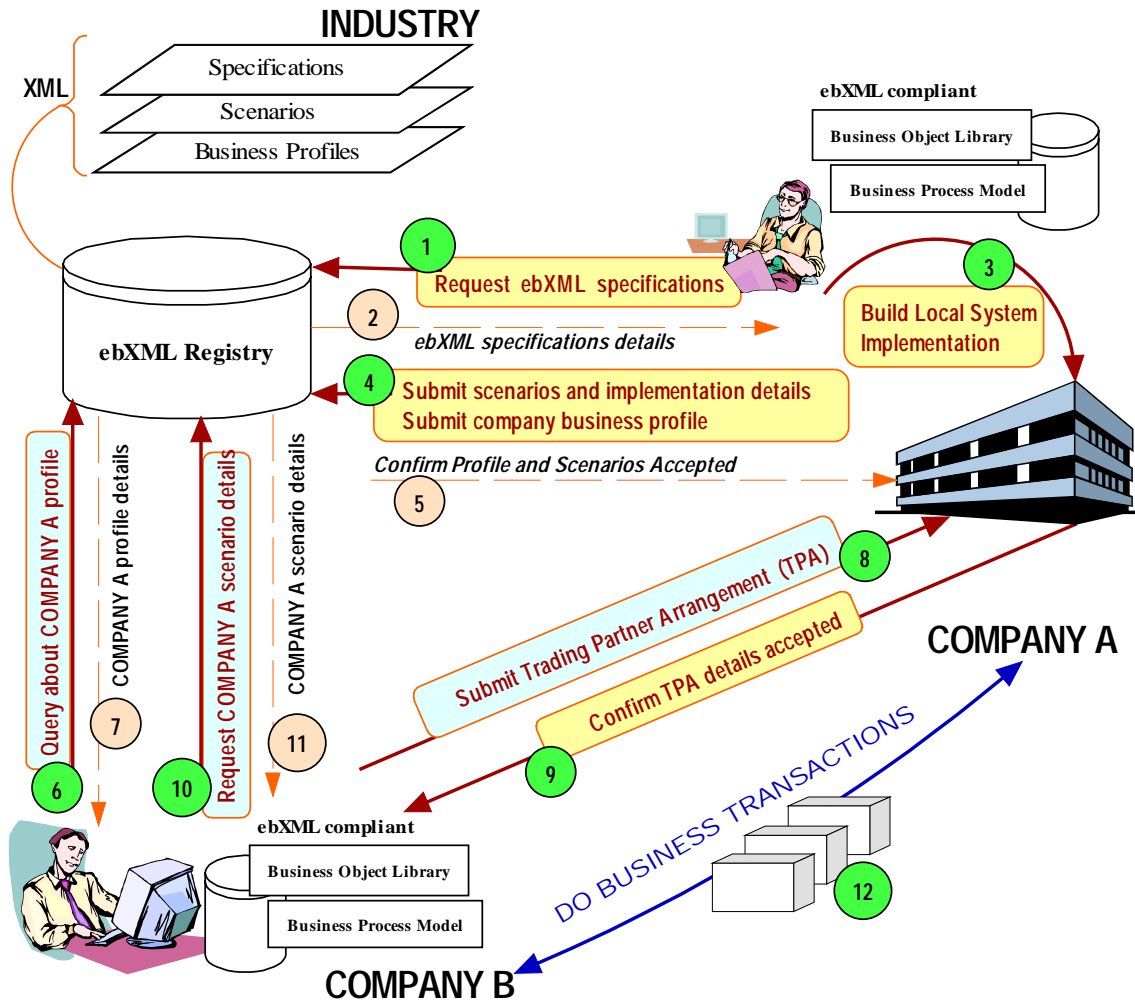
Figure 1: *a high level overview of ebXML functionality*

214

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

218   request results in the ebXML process specification being sent to Company A (Figure 1,
219   step 2). Company A, after reviewing the specification, decides to build and deploy its
220   own ebXML compliant application (Figure 1, step 3). [Note: custom software
221   development is not a necessary prerequisite for ebXML participation, user applications
222   will be also commercially available as turn-key solutions.]
223
224   Company A then submits its own implementation details, reference links, and *Trading*
225   *Partner Profile* (*TPP*) as a request to the ebXML registry (Figure 1, step 4). The *TPP*
226   submitted describes the company's ebXML capabilities and constraints, as well as its
227   supported business scenarios. These scenarios are *XML* versions of the *Business*
228   *Processes* and associated information parcels (based on business objects: for example a
229   sales tax calculation) that the company is able to engage in. After receiving verification
230   that the format and usage of a business object is correct, an acknowledgment is sent to
231   Company A by the ebXML Registry (Figure 1, step 5).
232
233   Company B (an SME) is then informed by Company A that they would like to engage in
234   a business transaction using ebXML. Company B acquires a shrink-wrapped application
235   that is ebXML compliant and able to interface with its existing (legacy) applications. The
236   ebXML program already contains the base ebXML information bundles such as a library
237   of *Business Object*s and Models for the specific industry they are part of. Company A
238   knows that its *Business Processes* and *TPP* are compliant with the ebXML infrastructure
239   from the information available in the ebXML specification package. However, since
240   Company A just registered its scenarios, they are not yet part of the package. Therefore
241   the ebXML application queries the ebXML Registry about Company A (Figure 1, step 6).
242   Company A's profile is retrieved (Figure 1, step 7). Based on the *TPP*, the application
243   determines that it is able to execute a specific scenario that Company A supports.
244
245   Before engaging in that the scenario Company B submits a proposed *Trading Partner*
246   Agreement (*TPA*) directly to Company A's ebXML compliant software interface. The
247   *TPA* outlines the *eBusiness* scenario and specific arrangement(s) it wants to use with
248   Company A, as well as certain messaging, contingency and security-related requirements
249   (Figure 1, step 8). Company A accepts the *TPA* and acknowledgement is sent directly to
250   Company B's shrink-wrapped ebXML software application (Figure 1, step 9). Since the
251   scenario from Company A was not available in the software package that Company B is
252   using, the application requests it from the ebXML Registry (Figure 1, step 10). The
253   scenario is then provided to Company B's application (Figure 1, step 11).
254
255   Based on the processes (contained in the process models) and information parcels
256   (presented in class diagrams) Company A and B are now engaging in *eBusiness* utilizing
257   ebXML specifications via their respective software applications (Figure 1, step 12).

258 The conceptual overview described in the scenario above introduced the following
259 concepts and architectural *Components*:
260
261   1. A standard mechanism for describing a *Business Process* and its associated
262   information model.
263   2. A mechanism for registering and storing a *Business Process* and information
264   model so that it can be shared/reused.
265   3. Discovery of information about each participant including:
266   • The *Business Processes* they support.
267   • The business service interfaces they offer in support of the *Business Process*.
268   • the business messages are to be exchanged between their respective service
269   interfaces.
270   • The technical configuration of the supported transport, security and encoding
271   protocols.
272   4. A mechanism for registering the aforementioned information so that it may be
273   discovered and retrieved.
274   5. A mechanism for describing a *Trading Partner* Agreement (*TPA*) which may be
275   derived from the information about each participant from item 3 above.
276   6. A standardized messaging service which enables interoperable, secure and
277   reliable exchange of messages between two parties.
278   7. A mechanism for configuration of the respective messaging services to engage in
279   the agreed upon *Business Process* in accordance with the constraints defined in
280   the *TPA*.
281
282 Using these *Components* ebXML compliant software can be used to implement popular,
283 well-known *eBusiness* scenarios, examples include but are not limited to:
284
285   a) Two partners set-up an agreement and run the associated electronic exchange.
286   b) Three or more partners set-up a *Business Process* implementing a supply-chain
287   and run the associated electronic exchanges
288   c) A company sets up a portal that defines a *Business Process* involving the use of
289   external business services.
290   d) Three or more parties engage in multi-party *Business Process* and run the
291   associated electronic exchanges.
292
293 The above examples are described in detail in Appendix A.

9.0 Relating the ebXML Architecture to Existing Standards

295
296 The ebXML approach utilizes public specifications and standards wherever applicable
297 and consistent with the goals of the ebXML initiative. One such specification is the
298 Open-edi work, an ISO/IEC 14662 (Open-edi Reference Model) vision of future *EDI*.
299 The ebXML approach can benefit from the lessons learned by Open-edi work and utilize
300 the related methodologies. Particularly, Open-edi takes a generic industry and technology
301 neutral approach and by similarly utilizing this, ebXML will enable organizations to

302   provide the opportunity to significantly lower the barriers to electronic data exchange by
303   introducing standard business scenarios and the necessary services to support them. In
304   principle, once a business scenario is agreed upon, and implementations conform to the
305   standards, there is no need for prior agreement among *Trading Partners*, other than the
306   decision to engage in the ebXML transaction in compliance with the business scenario.
307   This will lead to the ability to establish short-term business relationships quickly and cost
308   effectively.
309
310   The field of application of ebXML is the electronic processing of *XML*-based business
311   transactions among autonomous multiple organizations within and across sectors (e.g.,
312   public, private, industrial, geographic). It includes business transactions that involve
313   multiple data types such as numbers, characters, images and sound. The Open-edi
314   Reference Model provides the standards required for the inter-working of organizations
315   through interconnected information technology systems, and is independent of specific
316   information technology (IT) implementations, business content or conventions, business
317   activities, and organizations.
318
319   The Open-edi Reference Model places existing *EDI* standards in perspective using two
320   views to describe the relevant aspects of business transactions: the Business Operational
321   View (BOV) and the Functional Service View (FSV). The ebXML Architecture uses
322   similar views of these definitions. The BOV expresses the users' requirements needed to
323   achieve the common business goal. The FSV describes how the BOV is actually
324   implemented using the selected technology.
325

Open-edi Reference Model
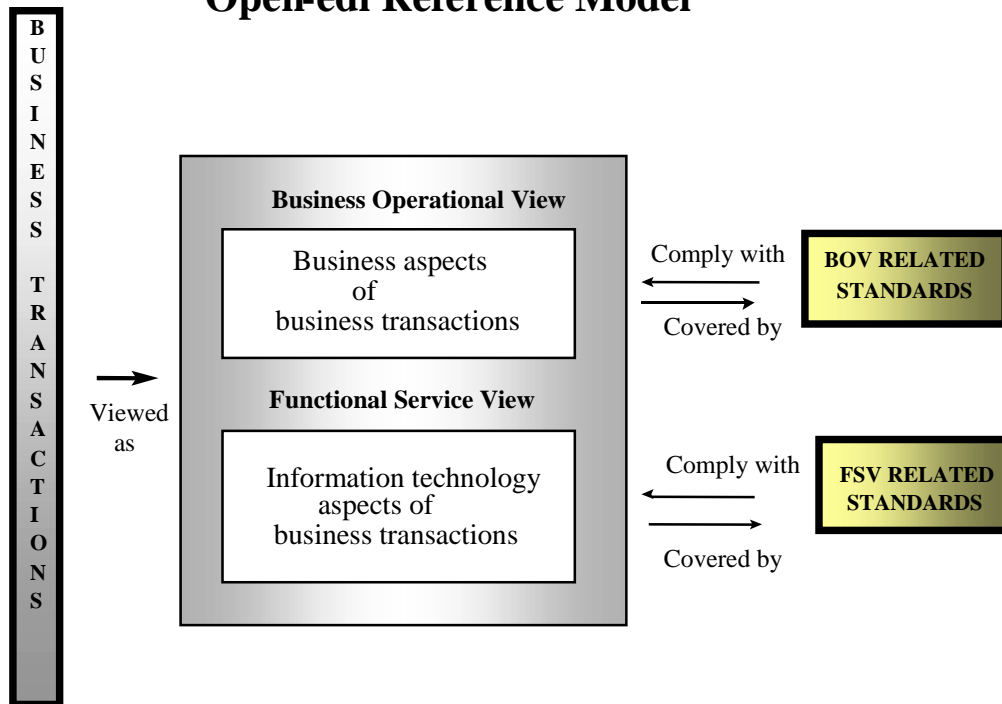
Figure 2: *Open-edi environment*

328

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

331   The primary focus of ebXML resides with the FSV and the supporting BOV. The
332   assumption for ebXML is that the FSV will be implemented by commercial software
333   vendors and ensure backwards compatibility to traditional *EDI* systems (where
334   applicable). As such, the resultant BOV-related standards provide the business and object
335   class models needed to construct ebXML compliant *eBusiness* services and *Components*.
336
337   While business practices from one business organization to another are highly variable,
338   most activities can be decomposed into *Business Processes* which are more generic to a
339   specific type of business. This analysis through the modeling process will identify object
340   classes and models that are likely candidates for standardization. The ebXML approach
341   looks for standard reusable *Components* from which to construct information exchange
342   software. While Open-edi is a theoretical syntax neutral approach, ebXML itself is
343   focused on a physical implementation using specifically an XML-based syntax and
344   related technologies.
345

346   *10.0 ebXML Architecture*

347

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

350

351   • The Business Operational View (BOV)
352   • The Functional Service View (FSV)

353

354   The BOV addresses the semantics of:

355

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

357

358   b) The architecture for business transactions, including:

359

360   o   operational conventions;
361   o   agreements;
362   o   mutual obligations and requirements.

363

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

365

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

368

369   • functional capabilities;
370   • service interfaces;

371   • protocols.
372
373   Additionally, the functional capabilities, service interfaces and protocols include:
374
375   • capabilities for implementation, discovery, deployment and run time scenarios;
376   • user application interfaces;
377   • data transfer infrastructure interfaces;
378   • protocols for interworking of XML vocabulary deployments from different
379   organizations.
380
381   The BOV and the FSV are discussed in detail in the following sections.
382

383   *11.0 ebXML Business Operational View*

384
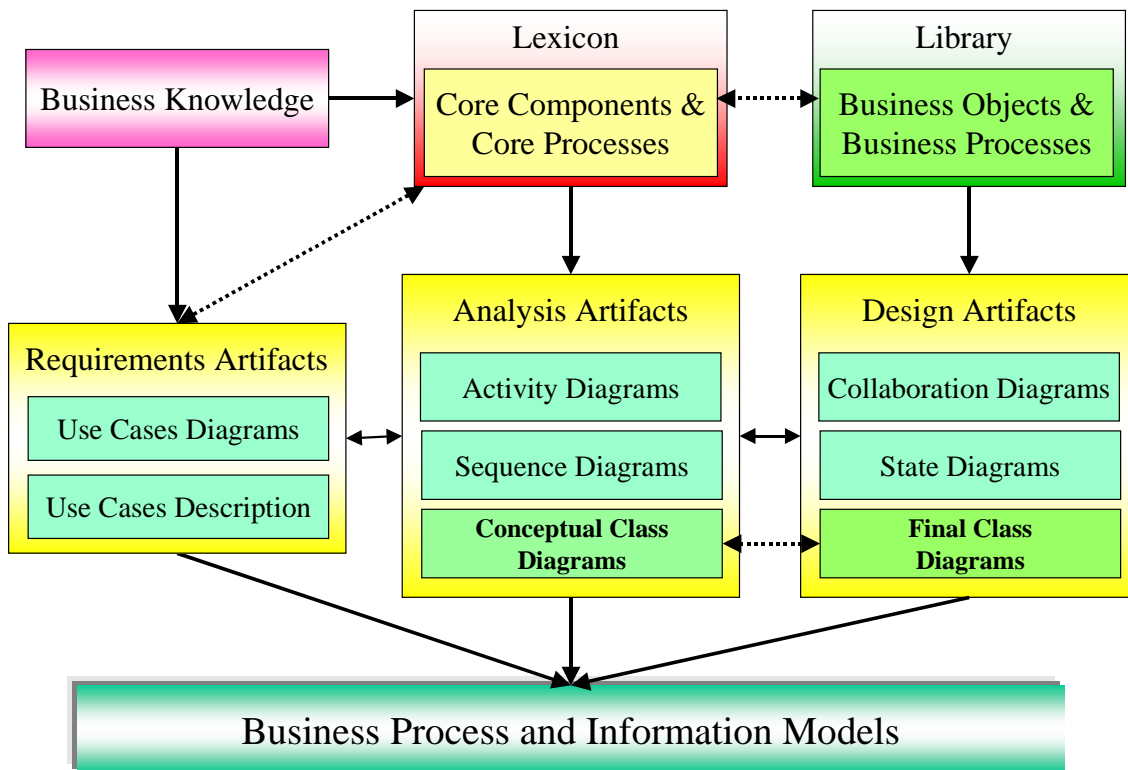
385
386
387   **Figure 3:** *the Business Operational View*
388
389   ebXML Business and Information Models are created following the selected ebXML
390   *Business Process and Information Modeling* (see section 17).
391
392   Business knowledge is captured in a *Lexicon*. The *Lexicon* contains data and process
393   definitions including relationships and cross-references as expressed in business
394   terminology and organized by industry domain. The *Lexicon* is the bridge between the

395   specific business or industry language and the knowledge expressed by the models in a
396   more generalized industry neutral language.
397
398   The first phase defines the requirements artifacts which describe the problem using Use
399   Case Diagrams and Descriptions. If *Lexicon* entries are available they will be utilized,
400   otherwise new *Lexicon* entries will be created.
401
402   The second phase (analysis) will create activity and sequence diagrams describing the
403   *Business Processes*. Class diagrams will capture the associated information parcels
404   (business messages). The analysis phase reflects the business knowledge contained in the
405   *Lexicon*. No effort is made to force the application of object-oriented principles. The
406   class diagram is a free structured data diagram.
407
408   The design phase is the last step of standardization, which may be accomplished by
409   applying object-oriented principles. In addition to generating collaboration diagrams, a
410   state diagram may also be created. The data diagram from the analysis phase will
411   undergo harmonization to align it with other models in the same industry and across
412   others.
413
414   Therefore in ebXML interoperability is achieved by applying *business object*s across all
415   class models. The content of the *business object library* is created by analyzing existing
416   *business object*s as used by many industries today in conjunction with the *Lexicon*
417   content and ebXML selected modeling methodology.
418
419   Figure 4 shows how the user can see this correlation to the actual business roles:
420
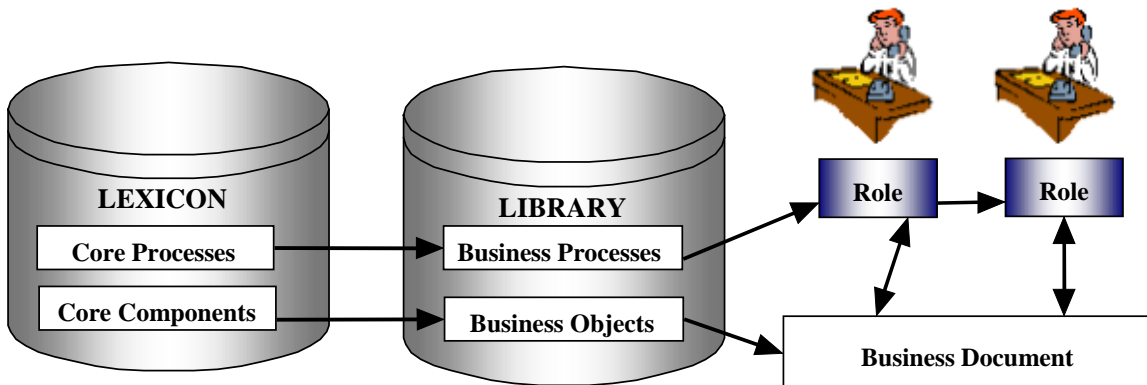
421
422
423   *Figure 4: Role Relation Model.*

424

12.0 ebXML Functional Service View

426

427
428
429   *Figure 5: ebXML Functional Service View*
430

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

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

446   ebXML Registry and Repository System. The latter is the preferred approach since it
447   provides a single access and maintenance and control point.
448
449   Additionally the UID keys may be implemented in physical *XML* syntax in a variety of
450   ways. The architectural needs require that several mechanisms be supported. These
451   mechanisms include, but are not limited to:
452
453   • A pure explicit reference mechanism (*XML* URN:UID method),
454   • A referential method (*XML* URI:UID / namespace:UID),
455   • An object-based reference compatible with W3C Schema (*XML*
456   URN:complextype name), and
457   • A datatype based reference (for ISO 8601:2000 Date/Time/Number datatyping
458   and then legacy datatyping).
459
460   Examples of each of these in *XML* syntax in the order noted include:
461
462   • An URN:UID method,
463   • An URI:UID / namespace:UID method,
464   • An URN:complextype name method, and
465   • An explicit type encoding values as outlined in ISO 8601.
466
467   Additionally, all participating *Components* in ebXML must facilitate multilingual
468   support. Again, a UID reference is particularly important here as it provides a language
469   neutral reference mechanism. To enable multilingual support, the ebXML specification
470   must be compliant with Unicode and ISO/IEC 10646 for character set and UTF-8 or
471   UTF-16 for character encoding.
472
473   The underlying ebXML Architecture is distributed in such a manner to minimize the
474   potential for a single point of failure within the ebXML infrastructure. This specifically
475   refers to Registry and Repository Services (see Registry and Repository Functionality,
476   Section 20 for details of this architecture).
477
478   The implementation of the FSV of ebXML, can be categorized as having the following
479   three major phases:
480
481   a) The Implementation Phase
482
483   The implementation phase deals specifically with the procedures for creating an
484   application of the ebXML infrastructure
485
486   b) The Discovery and Deployment Phase
487
488   The Discovery and Deployment Phase covers all aspects of actual discovery of
489   ebXML related resources and self enabled into the ebXML infrastructure.
490

491   c) The Run Time Phase
492
493   The Run Time phase covers the execution of a ebXML scenario with the actual
494   associated ebXML transactions.
495
496   These three phases are now discussed in greater detail.

497

13.0 Implementation Phase

499

500 A *Trading Partner* wishing to engage in an ebXML compliant transaction, must first
501 request a copy of the ebXML specification. The Specification is then downloaded to the
502 *Trading Partner*. The *Trading Partner* studies the ebXML specification. The *Trading*
503 *Partner* subsequently requests to download the *Lexicon* and the *Business Object Library*.
504 The *Trading Partner* may also request other *Trading Partners' Business Process*
505 information (stored in its *TPP*) for analysis and review. The *Trading Partner* may also
506 submit its own *Business Process* information to an ebXML compliant Registry.
507
508   *Figure 6 below, illustrates a potential interaction between an ebXML Registry and a business service*
509   *interface.*
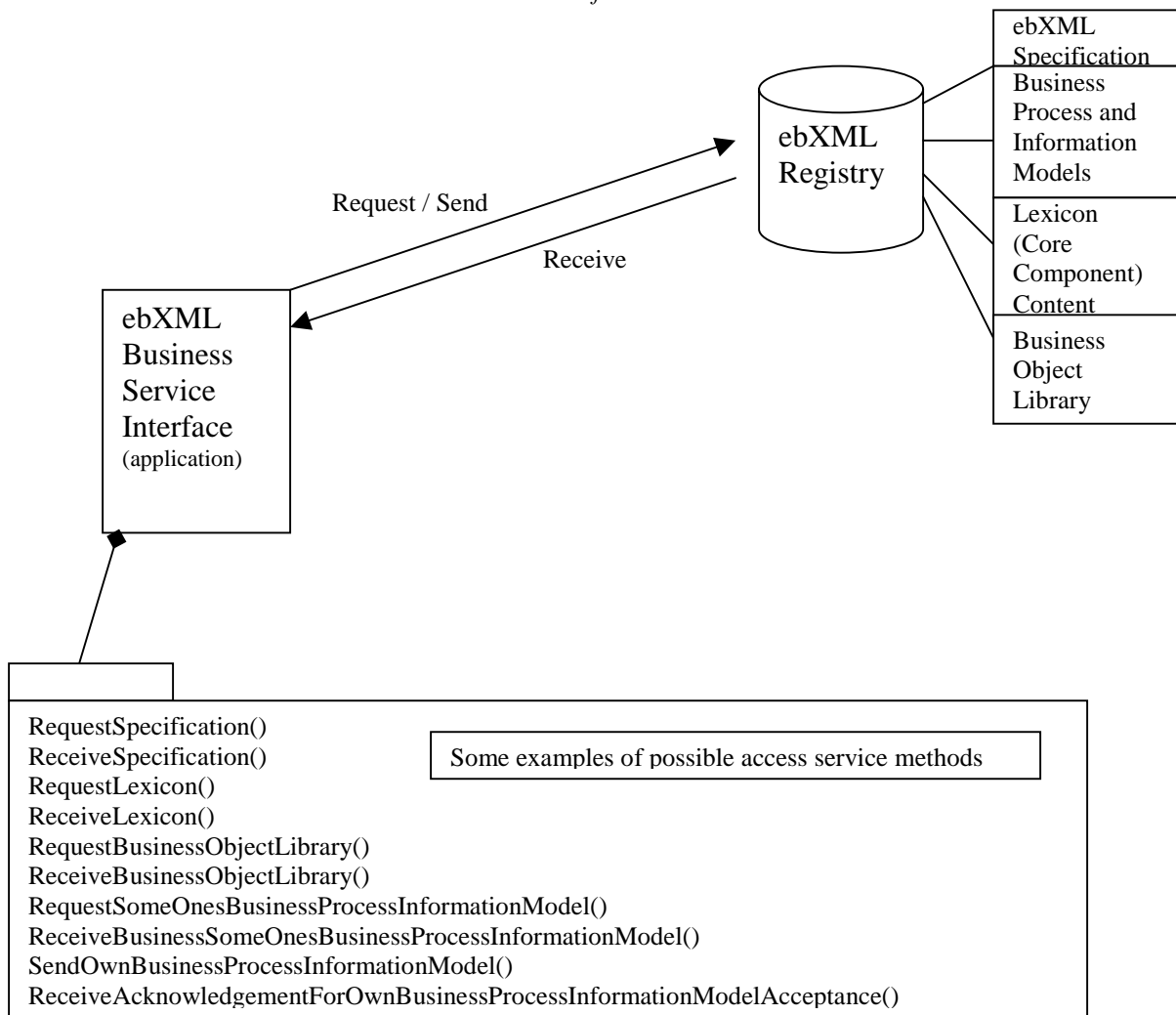
510   **Figure 6:** *Functional Service View: Implementation Phase*

511

512

513

514   *14.0 Discovery and Deployment Phase*

515

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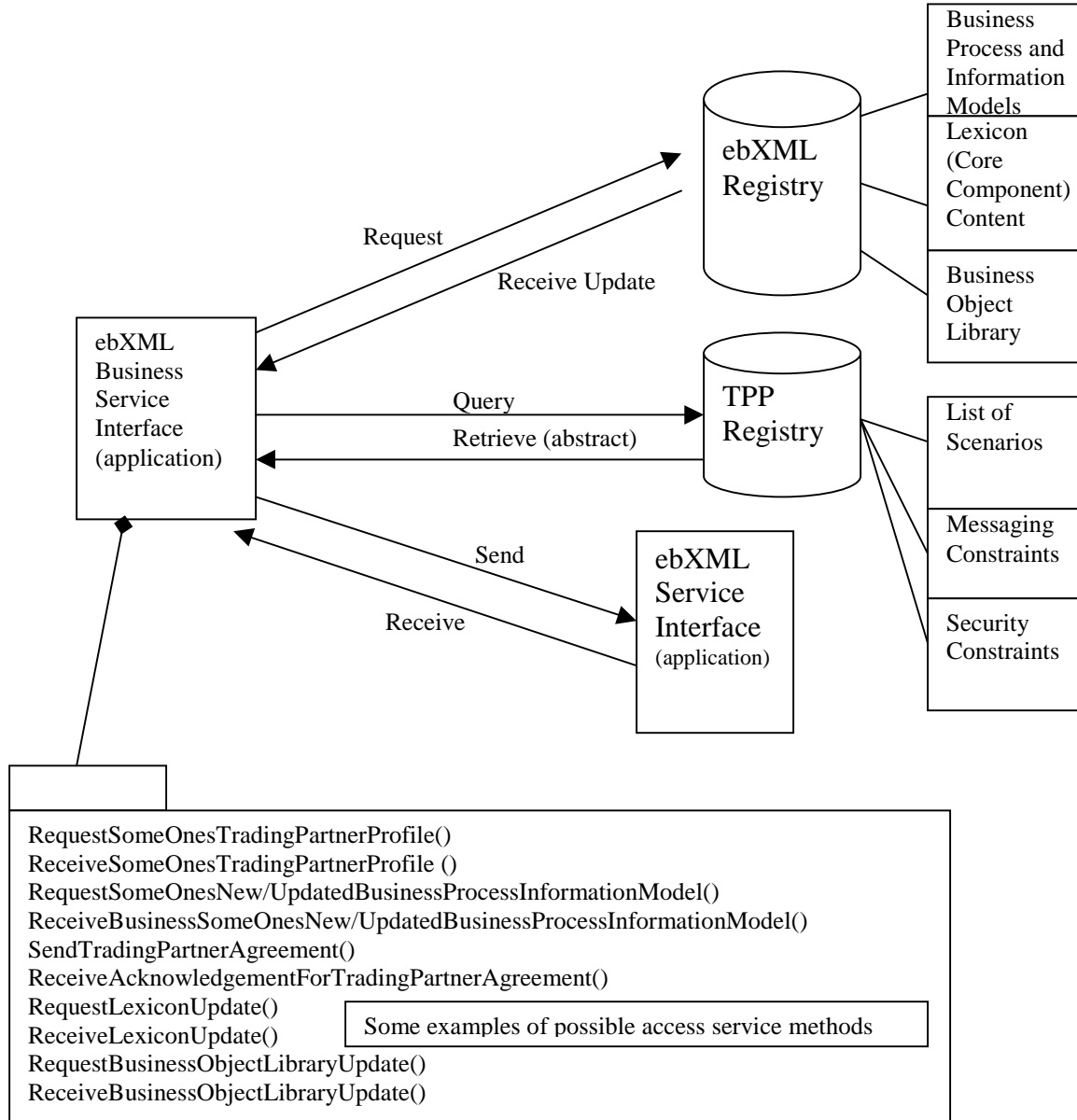
523
524   **Figure 7:** *Functional Service View: Discovery and Deployment Phase*

525   ## 15.0 Run Time Phase
526
527   The Run Time phase is the least complex (Figure 8). Note that no Registry calls are
528   required during the Run Time Phase. There are ebXML message instances being sent
529   and received between *Trading Partners* utilizing the ebXML Messaging Service.
530
531
532
533
534
535
536
537
538
539
540

541

542

543
544
545
546
547
548
549   **Figure 8:** *Functional Service View: Run Time Phase*
550

ebXML
Business
Service
Interface
(application)

Send
Retrieve
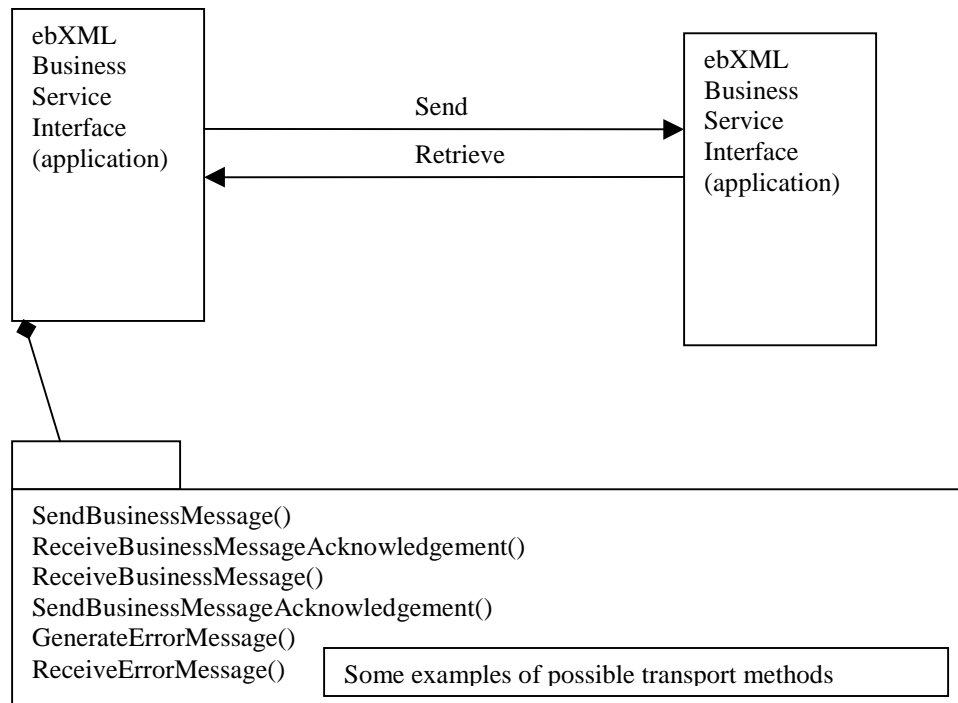
ebXML
Business
Service
Interface
(application)

SendBusinessMessage()
ReceiveBusinessMessageAcknowledgement()
ReceiveBusinessMessage()
SendBusinessMessageAcknowledgement()
GenerateErrorMessage()
ReceiveErrorMessage()

Some examples of possible transport methods

551   ## 16.0 Trading Partner Information
552
553   To facilitate the process of conducting *eBusiness*, *SMEs* and other organizations need a
554   mechanism to publish information about the *Business Processes* they support along with
555   specific technology implementation details about their capabilities for exchanging
556   business information. This is accomplished by creating a *Trading Partner Profile (TPP)*.
557   The TPP is a document which allows a *Trading Partner* to express their minimum
558   *Business Process* and Business Service Interface requirements in a manner where they
559   can be universally understood by other ebXML compliant *Trading Partners*. The TPP
560   describes the specific technology capabilities that a *Trading Partner* supports and the
561   Service Interface requirements that need to be met in order to exchange business
562   documents with that *Trading Partner*. The TPP of the a priori interchange information is
563   stored in an ebXML Registry which provides a discovery mechanism for *Trading*
564   *Partners* to find one another.
565

566   **16.1 Support for Trading Partner Agreements**
567   To facilitate the process of conducting electronic business, organizations need a
568   mechanism to publish information about the *Business Processes* they support, along with
569   specific technology details about their capabilities for sending and receiving business
570   documents.
571   ebXML defines the ability for this to be realized under the broad notion of a *Trading*
572   *Partner Agreement*.
573
574   A *Trading Partner Agreement* (*TPA*) is a document that describes: (1) the Messaging
575   Service (technology), and (2) the Process (application) requirements that are agreed upon
576   by two or more parties. A *TPA* is negotiated after the discovery process and is essentially
577   a snapshot of the specific technology and process related information that two or more
578   parties agree to use to exchange business information. If any of the parameters of an
579   accepted *TPA* changes after the agreement has been executed, a new *TPA* shall be
580   negotiated between all parties.
581
582   Conceptually, ebXML supports a three level view of narrowing subsets to arrive at
583   agreements for transacting business. The outer-most scope relates to all of the
584   possibilities that a Partner could do, with a subset of that of what a Partner is capable of
585   doing, with a subset of what a Partner "will" do.
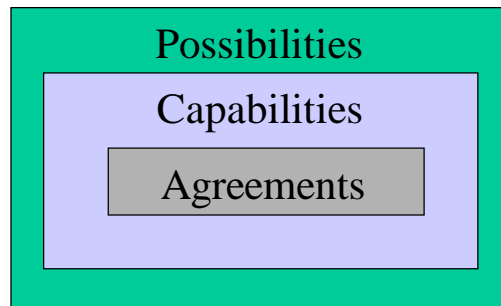
586
587   ***Figure 9:*** *Three level view of TPA's*
588
589   ebXML acknowledges the global scope of a *Trading Partner Agreement*s to include such
590   aspects as legal agreement elements and legal ramifications and other trade issues that
591   are, from an over-arching business perspective, essential elements of "Agreements
592   between Traders." ebXML limits its scope within this broad spectrum to addressing the
593   needs of (electronic) *Business Collaborations*. This provides extensibility for ebXML to
594   expand to encompass other aspects of *Trading Partner Agreement*s on its own or by
595   embracing other work. Further, the entities engaged in Business Collaborations within
596   ebXML are referred to as *Partners*. Business Collaborations are the first order of support
597   that can be claimed by ebXML Partners. This "claiming of support" for specific Business
598   Collaborations is facilitated by a distinct profile defined specifically for publishing, or
599   advertising in a directory service, like the ebXML Registry/Repository or other available
600   similar services.
601
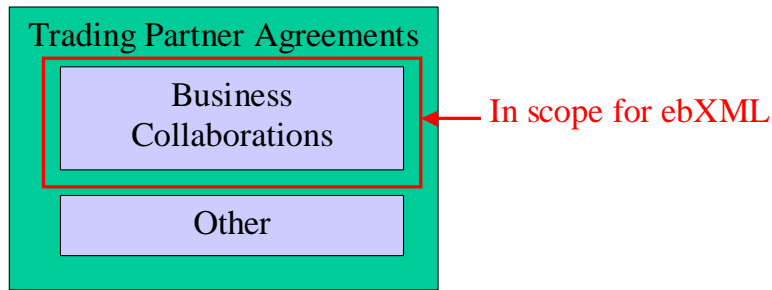
Figure 10: *Scope for TPA's*

17.0 Business Process and Information Modeling

17.1 Overview

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

17.2 Position within overall ebXML Architecture

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

626
17.3 Business Process and Information Modeling Functionality
628 The *Business Process and Information Modeling* specification supports *UML Business*
629 *Process* and information modeling, along with conversion of *UML* models into *XML*, and
630 direct access to the *XML* expression of the model. Within the *UML* modeling system the
631 ebXML specification provides a *UML* profile, a set of recommended diagrams, and a
632 selected methodology to follow in constructing those diagrams. For the conversion of
633 *UML* to *XML* the specification provides a set of production rules. For further
634 standardization, the specification provides a set of core processes, and a set of patterns
635 from which to compose new process definitions.
636
637 The ebXML Metamodel specifications constitute a set of *XML* structures that can be
638 populated and stored in an ebXML Registry and Repository System. The *XML* structures
639 may utilize a classification system with UID reference linkages which are compatible
640 with the Registry and Repository architecture requirements.
641
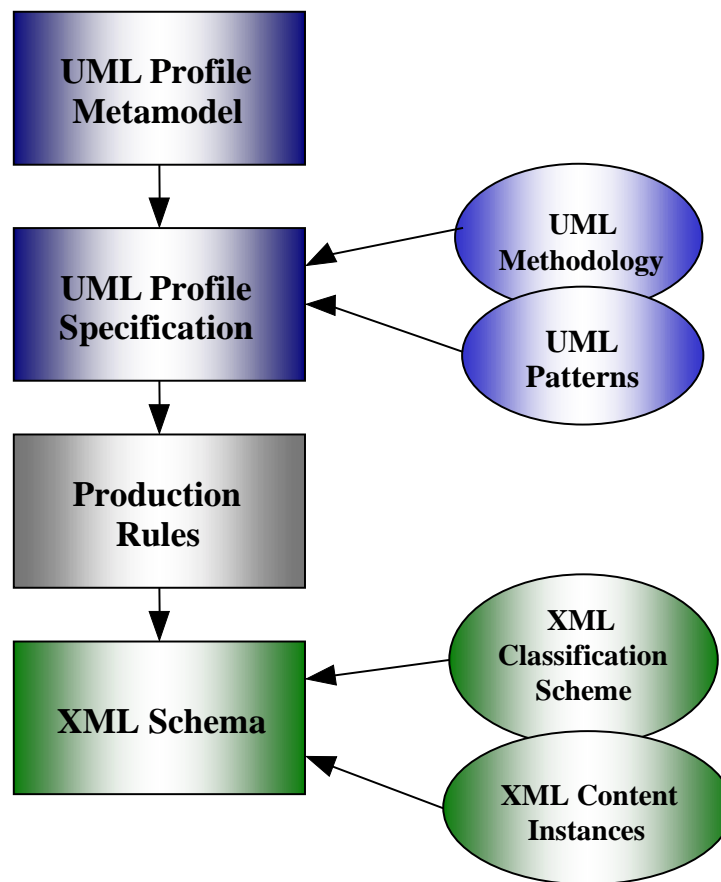
642
643
644   ***Figure 11:*** *Relationship of UML to ebXML metamodel content representation.*
645
17.4 The Business Process and Information Metamodel
647 The *Business Process* and Information Metamodel is composed of four layers (see Figure
648 12 below). The top layer consists of the Business Operational Map which supports the

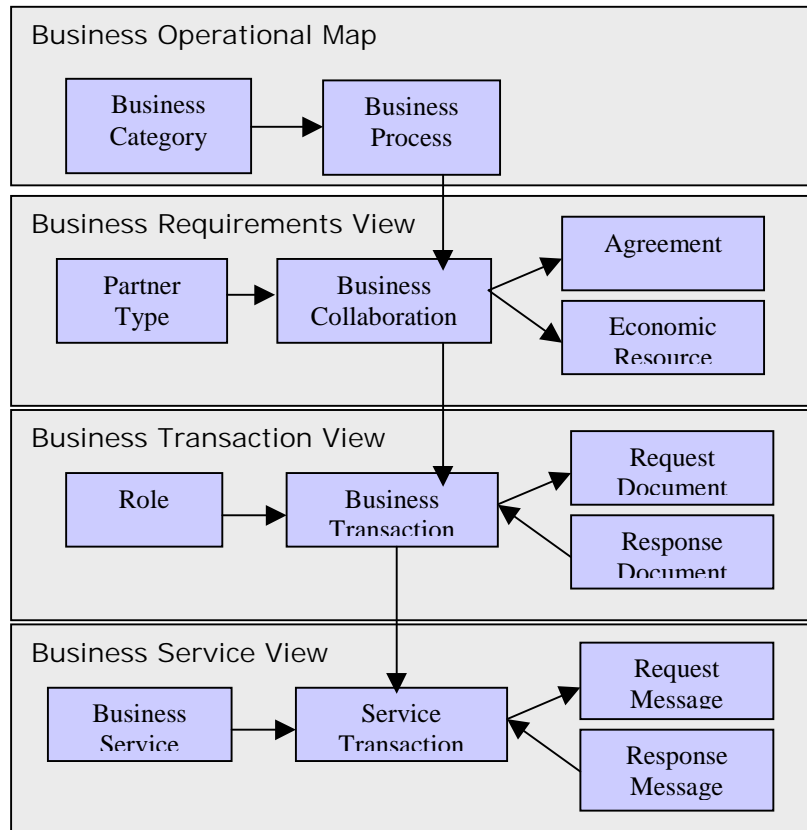649   process of relating different *Business Processes* to each other into a map as well as the
650   categorization of *Business Processes* by business or process area. The next layer, the
651   Business Requirements View, supports definition of the partner type which partakes in a
652   step within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

658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681

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

683
684   ## 17.5 Interfaces and Relationship
685   The interface from a *Business Process* and information model to other parts of the
686   ebXML architecture, or to other tools and environments outside the scope of the ebXML
687   specifications, is an *XML* document representing the *Business Process* and information
688   model. Specifically, the interface between the *Business Process* and information model
689   and the *Trading Partner* model is the part of such an *XML* document that represents the
690   business transactional layer of the *Business Process* metamodel. The expression of the
691   sequence of commercial transactions in *XML* is shared between the *Business Process* and
692   *Trading Partner* models.
693

694   **17.6 Relationship to Trading Partner Agreements**
695   The interface between the *Business Process* and information model and the *Trading*
696   *Partner* specification is the sequence of business transactions, the commercial business
697   itself, and the message exchange in support of the business transaction. The profile of a
698   *Trading Partner* defines that partner's functional and technical capability to support one
699   or more roles in a *Business Process*. The agreement between two *Trading Partners*
700   defines the actual conditions under which the two partners will conduct business
701   transactions together.
702
703   **17.7 Relationship to Core Components**
704   A *Business Process* can be seen as a series of actions on entities within an enterprise,
705   interleaved with a set of communications with parties outside the enterprise. The
706   communication between the parties is the shared part of the *Business Process*. This is the
707   focus of ebXML.
708
709   The entities within an enterprise are called business entities, and their data structure can
710   be represented by *Business Object*s.
711
712   The communication with parties outside the enterprise takes place through an exchange
713   of business documents.
714
715   Both *Business Object*s and business documents are composed from *Core Components*, re-
716   useable low-level data structures.
717
718   The exact composition of a *Business Object* or a business document is guided by a set of
719   contexts derived from (among other sources) the *Business Process*.
720

721   ## 18.0 Core Component Functionality

722
723   A *Core Component* captures information about a real world (business) concept, and
724   relationships between that concept and other business concepts.
725
726   A *Core Component* can be either an individual piece of business information, or a natural
727   "go-together" family of business information pieces. It is 'Core' because it occurs in
728   many different areas of industry/business information interaction.
729
730   A *Core Component* may contain:
731   • Another *Core Component* in combination with one or more individual business
732   information pieces.
733   • Other *Core Components* in combination with zero or more individual business
734   information pieces.
735

736   A *Core Component* needs to contain either attribute(s) or be part of another *Core*
737   *Component*, thus specifying the precise context or combination of contexts in which it is
738   used.
739
740   Context may be structural, identifying the placement of a *Core Component* within
741   another *Core Component*. It may be a combination of structural contexts when the *Core*
742   *Component* is re-used at different layers within another *Core Component*.
743
744   Context will also be defined by the *Business Process* model, which defines the instances
745   in which the *Business Object* occurs.
746
747
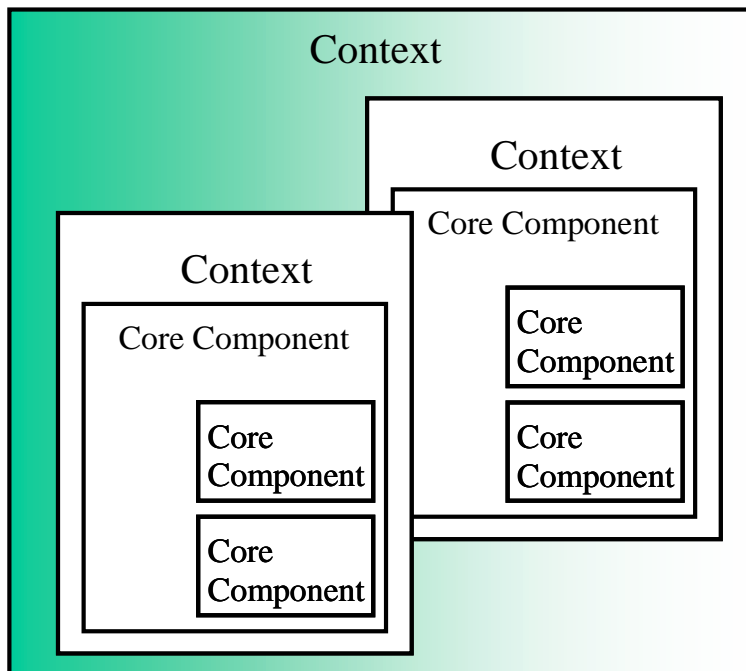
748
749
750   ***Figure 13:*** *Core components as contextual items.*
751
752   The pieces of information, or *Core Components*, within a generic *Core Component* may
753   be either mandatory, or optional. A *Core Component* in a specific context or combination
754   of contexts may alter the fundamental mandatory/optional cardinality.
755
756   Individual *Core Components* will in general match the "data list" part of *Business*
757   *Objects*.

758   ## 19.0 Business Object Functionality

759
760   ### 19.1 Overview
761   The term *Business Object* is used in two distinct ways in ebXML, with different
762   meanings for each usage:

763
764   • In a business model, *Business Objects* describe a business itself, and its business
765   context. The *Business Objects* capture business concepts and expresses an abstract
766   view of the business's "real world" functions.
767
768   • In a business software application or service, *Business Objects* reflects how
769   business concepts are represented in software. The abstraction here reflects the
770   transformation of business ideas (processes) into a software implementation.
771
772 Within the context of ebXML, only *Business Objects* represented in *Business Processes*
773 and information models are of relevance.
774

19.2 Business Objects in Business Process and Information Models

776 A *Business Object* describes a thing, concept, process or event in operation, management,
777 planning or accounting of a business or other organization. It is a conceptual object that
778 has been specified for the purpose of directly describing and representing, and thus
779 serving, a business concept or purpose. The focus/subject is the business subject/concept
780 being modeled.
781
782 A *Business Object* in this usage is a specification for a kind of object which may exist in
783 one or more business domains. The specification of a business object may include
784 attributes, relationships, and actions/events that apply to these objects. These business
785 object models may exist regardless of the existence of information systems, applications,
786 software design or program code. They are independent of information systems because
787 business object models directly reflect and abstract "real world" business concepts and
788 scenarios. Thus business object models are defined independently of application systems.
789
790 The primary concern when creating business object models is capturing common
791 business semantics and having a common idea or concept that is usable by different parts
792 of a business or by different independent businesses.
793

19.3 Common Business Objects

795 A *Common Business Object* (CBO) is a business object that is specified in more than one
796 Domain. For the purposes of defining CBOs, a domain is defined as an industry sector.
797 As with all business objects in general, the most important issue with CBOs is a common
798 concept and mutually agreed upon structure.

799   ## *20.0 Registry and Repository Functionality*

800
801   ### 20.1 Overview
802   An ebXML Registry provides a set of distributed services that enable the sharing of
803   information between interested parties for the purpose of enabling *Business Process*
804   integration between such parties by utilizing the ebXML specifications. The shared
805   information is maintained as objects in an ebXML Repository which is managed by
806   ebXML Registry Services. Access to an ebXML Repository is provided by the interfaces
807   (APIs) exposed by Registry Services.
808
809   Therefore, architecturally the Registry and Repository are tightly coupled *Components*.
810   The Registry provides the access services interfacing, the information model and
811   reference system implementation, while a Repository provides the physical backend
812   information store. For example, an ebXML Registry may provide a *Trading Partner*
813   *Profile* from the Repository in response to a query; or an ebXML Repository may contain
814   reference DTD's or Schemas that are retrieved by the Registry as a result of searching a
815   metadata classification of the DTD's or Schemas. Figure 14 provides an overview of this
816   configuration.
817
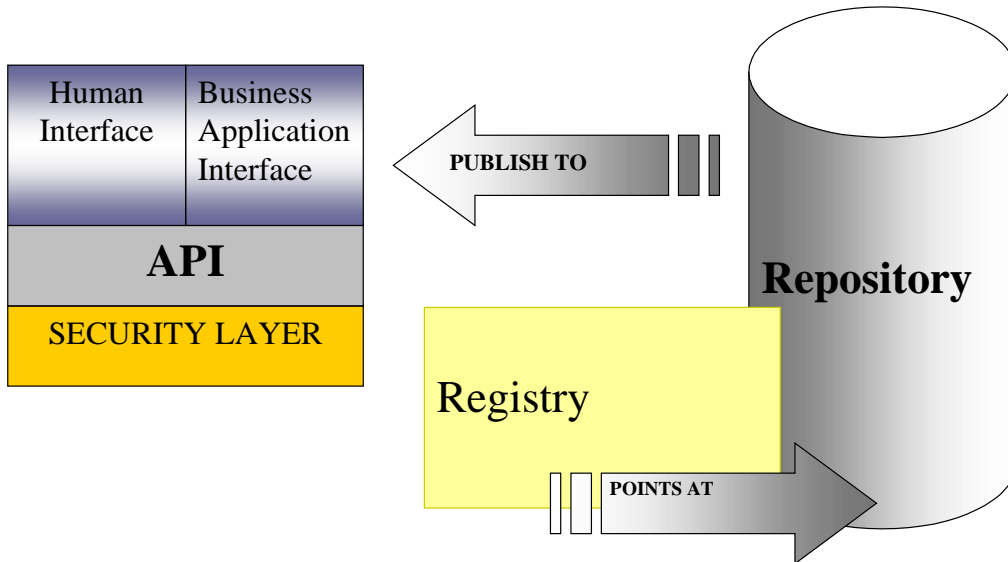
818
819   **Figure 14:** *Registry / Repository interaction overview.*
820
821
822

823 ## 20.2 Information Model & Interface Constrains

824 In order to accurately and consistently store and retrieve information a registry requires a
825 formal architecture that includes an information model. Traditional relational SQL
826 databases have a very simple information model that includes tables, indexes and column
827 definitions for content to be stored into them. However, when using *XML* structures to
828 store and manage content we potentially have an infinite variety of ways to present
829 information to a registry. We therefore to ensure there are particular aspects of those
830 structures that allow us to manage them within the registry, and that also these aspects are
831 linked to the access methods that will be used to interface to the registry. Providing the
832 mechanisms to support the business functional capabilities expressed in these *XML*
833 structures and content and ensuring it functions correctly is the role of the information
834 model.
835
836 The information model for the ebXML Registry is an extension of the existing OASIS
837 Registry information model, specifically tailored for the storage and retrieval of business
838 information content, whereas the OASIS model is a superset designed for handling
839 extended and generic information content. As such the ebXML Registry information
840 model is designed to make it easier to implement and to provide explicit ebXML
841 metamodel compliant instance structures to facilitate accessing and storing ebXML
842 content.
843
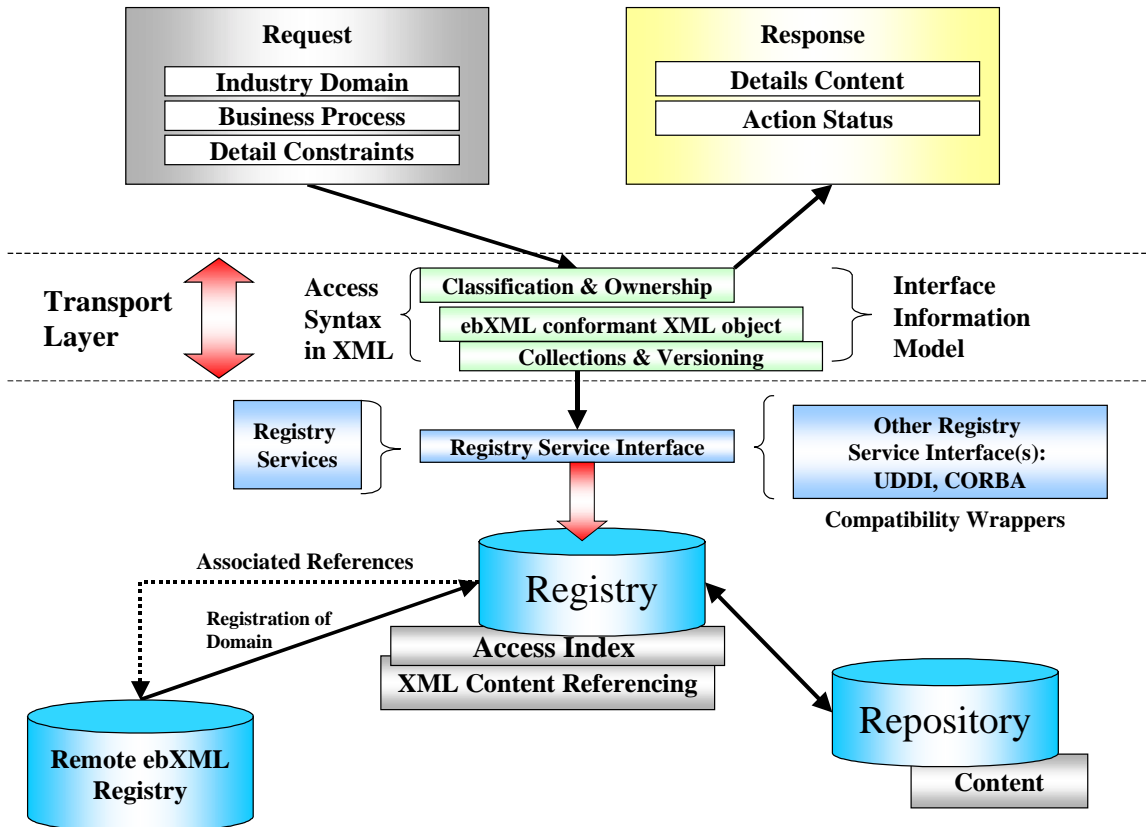
844
845
846 *Figure 15: Registry / Repository Architecture.*

847
848   A Registry maintains the metadata for a registered object, and a Repository maintains the
849   file containing a registered object. The Registry and Repository are tied together in that
850   the metadata for a registered object in the Registry includes a globally unique locator for
851   a file, in some Repository, that contains the registered object.
852
853   A Registry Item contains information that identifies, names, and describes each registered
854   object, gives its administrative and access status, defines its persistence and mutability,
855   classifies it according to pre-defined classification categories, declares its file
856   representation type, and identifies the submitting and responsible organizations.
857
858   Related to this the existing ISO11179/3 work on business semantic content registry
859   implementations is used to provide a model for the ebXML Registry implementation.
860   Again the approach is to take a tailored subset of the ISO11179 functionality that is
861   applicable to the ebXML Registry requirements, and to make implementing ebXML
862   Registry systems simpler for vendors to implement and librarians to manage.
863
864   Combined together these reference specifications are then exposed via the Registry
865   Interface system itself. The Registry interface is the architectural component that
866   provides both a machine-to-registry automated access system, and also a human-to-
867   registry interactive visual access system.   The Registry interface system is designed to be
868   a primitive *XML*-based interface that is transport layer neutral. However, the reference
869   implementation of the Registry interface is built using the ebXML Transport layer
870   facilities only.   Similarly the query syntax used by the Registry access mechanisms is
871   designed to be a neutral syntax based solely in *XML* syntax, and independent of the
872   physical product implementation of the backend Repository system.
873
874   ## 20.3 Formal Functional Overview
875   A Registry/Repository system may have many deployment models that yield the same
876   functionality. The initial specification and implementation will define the minimal
877   functional requirements that a Registry/Repository System shall provide to facilitate its
878   role in the ebXML infrastructure. It is expected that future specifications and
879   implementations will evolve into more complex systems.
880
881   All interaction between a Registry clients and the Registry are treated as business
882   transactions between parties. Thus the processes supported by the Registry are described
883   in terms of:
884
885   • A special *TPA* between the Registry and Registry clients.
886   • A set of business functional processes involving the Registry and Registry clients.
887   • A set of business messages exchanged between a Registry client and the Registry
888   as part of a specific business functional process.
889   • A set of primitive interface mechanisms to support the business messages and
890   associated query and response mechanisms.
891   • A special *TPA* for between one Registry interoperating with another Registry.

892   • A set of functional processes involving Registry to Registry interactions.
893   • A set of error responses and conditions with remedial actions.
894
895   The Registry interactions supported here are intended to be a limited subset of the full
896   requirements as defined by the ebXML Requirements documents. The architecture
897   described here is based on supporting the conceptual ebXML architecture and business
898   interactions as defined in Section 8 of this specification. Some of the extended
899   functionality deferred to a subsequent phase includes transformation services, workflow
900   services, quality assurance services and extended security mechanisms.
901

902 20.4 Sample Objects Residing in a Repository and Managed by a Registry

903   • **Schema**: These objects are documents that represent the schema (*XML* DTD, etc.)
904   for *XML* documents.
905

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

911   • *Trading Partner Profile*: These are *XML* documents that provide information
912   about a party interested in participating in B2B interaction.
913

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

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

926 20.5 Registry Management of Repository Objects and Metadata

927   Registry messages shall exist to create, modify and delete Repository objects and their
928   metadata. Appropriate security protocols shall be deployed to offer authentication and
929   protection for the Repository when accessed by the Registry.
930
931   Additionally all content stored into a Registry/Repository is implicitly public and open
932   information. Therefore parties submitting information to an ebXML Registry should
933   ensure that they have appropriate intellectual rights and permissions to submit this
934   information. An ebXML Registry will provide administrative access rights to ensure only
935   the submitting organization has formal access to change the content, however all other
936   retrieval rights will be open. For this reason, *TPA*s, which are necessarily proprietary to

937 *Trading Partners* will not be stored within an ebXML Registry, only the public *TPP*
938 details will be stored within an ebXML Registry.
939

940 20.6 Querying Registries and Returning Repository Objects and Metadata

941 A Registry query mechanism shall be employed to query for Repository objects and their
942 metadata by either an Application automated interface or a Human software GUI
943 interface.
944

945 Repository objects and their metadata shall be made available by ebXML messages sent
946 to the Registry (typically an Application requestor service).
947

948 Repository objects and their metadata can also be addressable where applicable as an
949 *XML* based URI reference using only HTTP for simple direct access.
950

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

954 Metadata queries perform an object search based on the metadata defined for (but
955 maintained outside) a managed object.
956

957 Browse and drill down queries are expected to be the primary use case for querying the
958 Registry by Web based human interactions. In this scenario, a user browses the repository
959 content using a Web browser via a HTTP protocol. The user may initially browse and
960 traverse the content based on the built-in classification schemes.
961

962 20.7 Registry to Registry Interfacing Model

963 Since ebXML Registries are distributed each Registry may potentially interact with and
964 cross-reference to another ebXML Registry. The following diagram provides an example
965 of the architectural *Components* that facilitate these mechanisms.
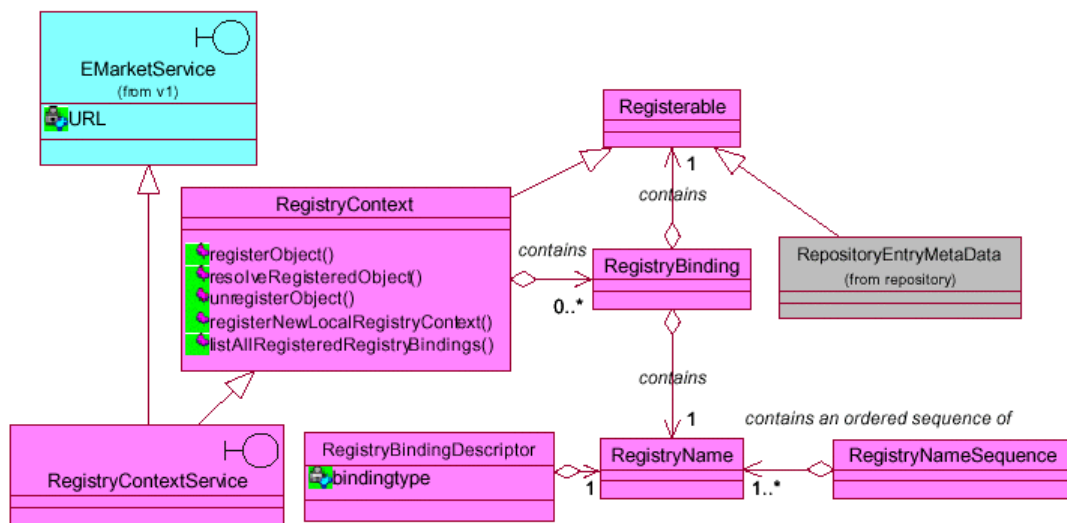
966
967   ***Figure 16:*** *Registry to Registry Service Components.*

968
969   Referencing Figure 16 above, the following apply:
970
971   A *RegistryName* to *RegistryObject* association is called a *RegistryBindin*g. A
972   RegistryBinding is always defined relative to a *RegistryContex*t. A *RegistryContext* is an
973   object that contains a set of RegistryBindings in which each RegistryName is unique.
974   Different RegistryNames can be bound to a RegisteredObject in the same or different
975   RegistryContexts at the same time.
976
977   A RegistryObject specialization (a RegistryContext or a RepositoryEntryMetaData),
978   which is bound into a RegistryContext is unaware of the fact that it has been associated to
979   a RegistryName via a RegistryBinding, and that the RegistryBinding may be bound into a
980   RegistryContext (not navigable).
981
982   A RegistryName is used to identify the binding within the RegistryContext for which it
983   may be bound. A *RegistryNameSequence* is an ordered set of RegistryNames that can be
984   used to resolve a RegisteredObject from a given target RegistryContext.
985   RegistryContextService is RegistryContext boundary interface and is an EMarketService.
986   For the extent of the model scope of this document, a URL is inherited and is used to
987   facilitate distribution of RegistryContexts through URL addressing.
988
989   A RegistryBindingDescriptor describes a RegistryBinding by identifying the type of
990   binding and the RegistryName. RegistryBindingDescriptors are returned on list messages
991   on RegistryContexts.
992
993   The architecture of the ebXML metadata classification system within the ebXML
994   Registry itself will be extended (see Figure 14 above). These extensions will support
995   references to domains that are not directly managed by that Registry, and its associated
996   Repository store.
997

998   **20.8 Registry/Repository Business Scenario Example**
999   In addition to the use of the ebXML Registry as a means to facilitate and enable the core
1000 architecture of ebXML compliant information exchanges, a Registry/Repository may also
1001 be used to facilitate business functional implementations. An example would be a
1002 network of *Trading Partners* similar to a telephone directory Yellow Pages system where
1003 businesses can be categorized by services that they provide.
1004
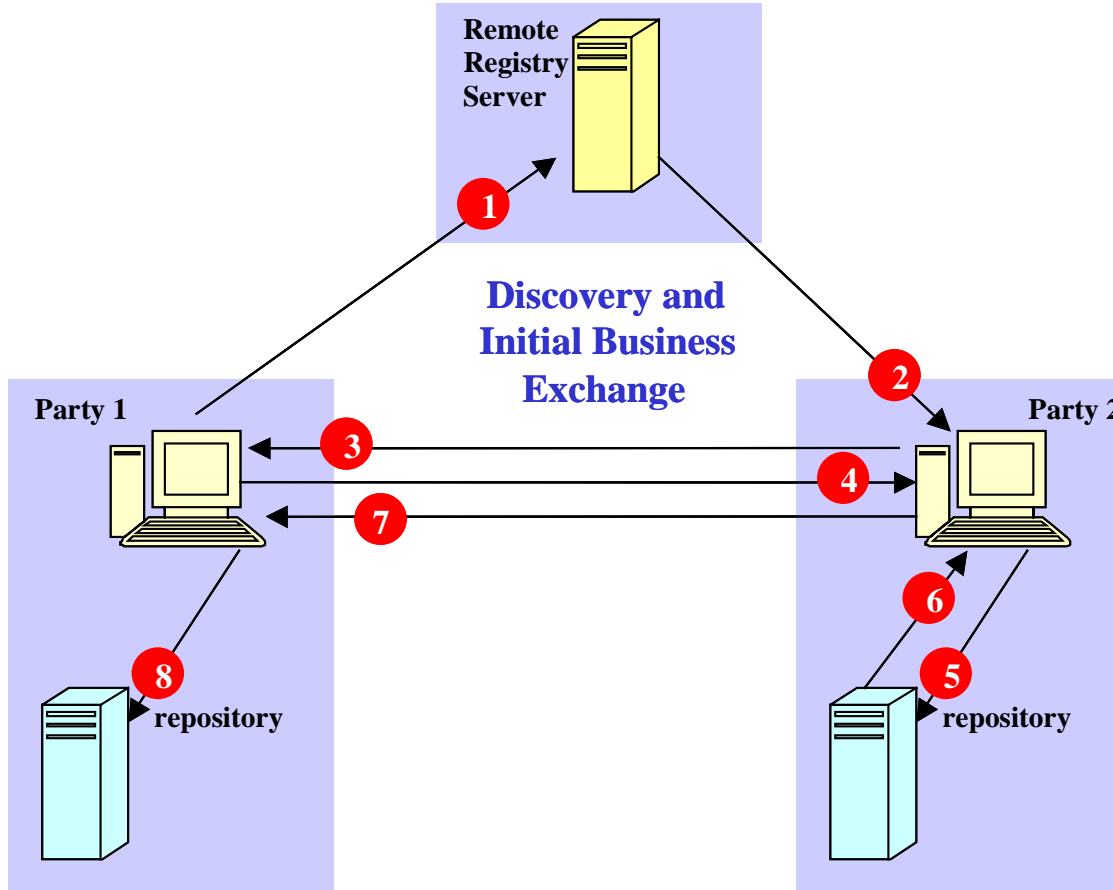
1005
1006   *Figure 17: Trading Partner Discovery.*
1007
1008

1009 ## *21.0 Messaging Service Functionality*

1010
1011 **21.1 Overview**
1012 The *ebXML Messaging Service* provides for the secure, reliable exchange of *ebXML*
1013 *Messages* between *Parties* over various transport protocols (SMTP, HTTP/S, FTP, etc.).
1014 The Messaging Service specification defines the MIME packaging and ebXML message
1015 *Header* information required by the *ebXML Messaging Service* to enable interoperable
1016 exchange of ebXML compliant messages.
1017
1018 The *ebXML Messaging Service* supports all messaging between distributed *Components*
1019 of the ebXML system including Registry/Repository and ebXML compliant applications.

1020   It utilizes and enforces the "rules of engagement" defined in a *Trading Partner*
1021   *Agreement (TPA).* The *ebXML Messaging Service* supports simplex (one-way) and
1022   request/response (either synchronous or asynchronous) message exchange and can be
1023   mapped onto any transport service capable of transporting MIME (further discussed in
1024   section).
1025
1026   The *ebXML Messaging Service* is conceptually broken down into three parts: (1) an
1027   abstract service interface, (2) functions provided by the Messaging Service Layer, and (3)
1028   the mapping to underlying transport service(s). The relation of the abstract interface,
1029   Messaging Service Layer, and transport service(s) are shown in Figure 18 below:
1030
1031
1032   ┌───┐
1033   │   Abstract ebXML Messaging Service Interface   │
1034   │   │
1035   │   EbXML Messaging Service Layer maps the   │
1036   │   abstract interface to the underlying transport │
1037   │   service   │
1038   │   │
1039   │   Transport Service(s)   │
1040   │   │
1041   └───┘
1042
1043   *Figure 18: ebXML Messaging Service*
1044
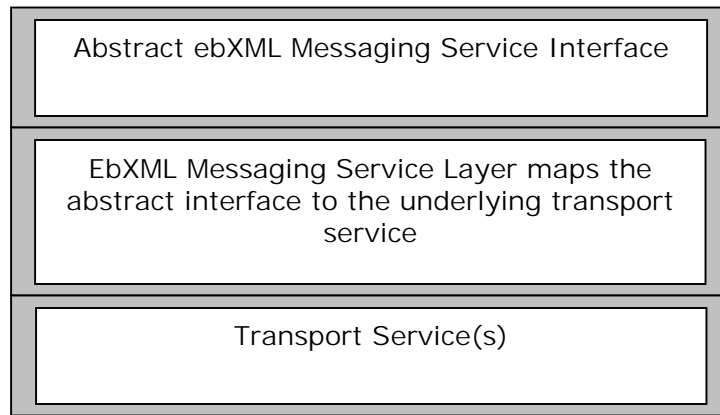
21.2 Abstract ebXML Messaging Service Interface

1046   The ebXML Message Service provides ebXML with an abstract interface whose
1047   functions, at an abstract level, include:
1048
1049   •   Send – send an *ebXML Message* – values for the parameters are derived from the
1050   *ebXML Message Headers*.
1051   •   Receive – indicates willingness to receive an *ebXML Message*.
1052   •   Notify – provides notification of expected and unexpected events.
1053   •   Inquire – provides a method of querying the status of the particular ebXML
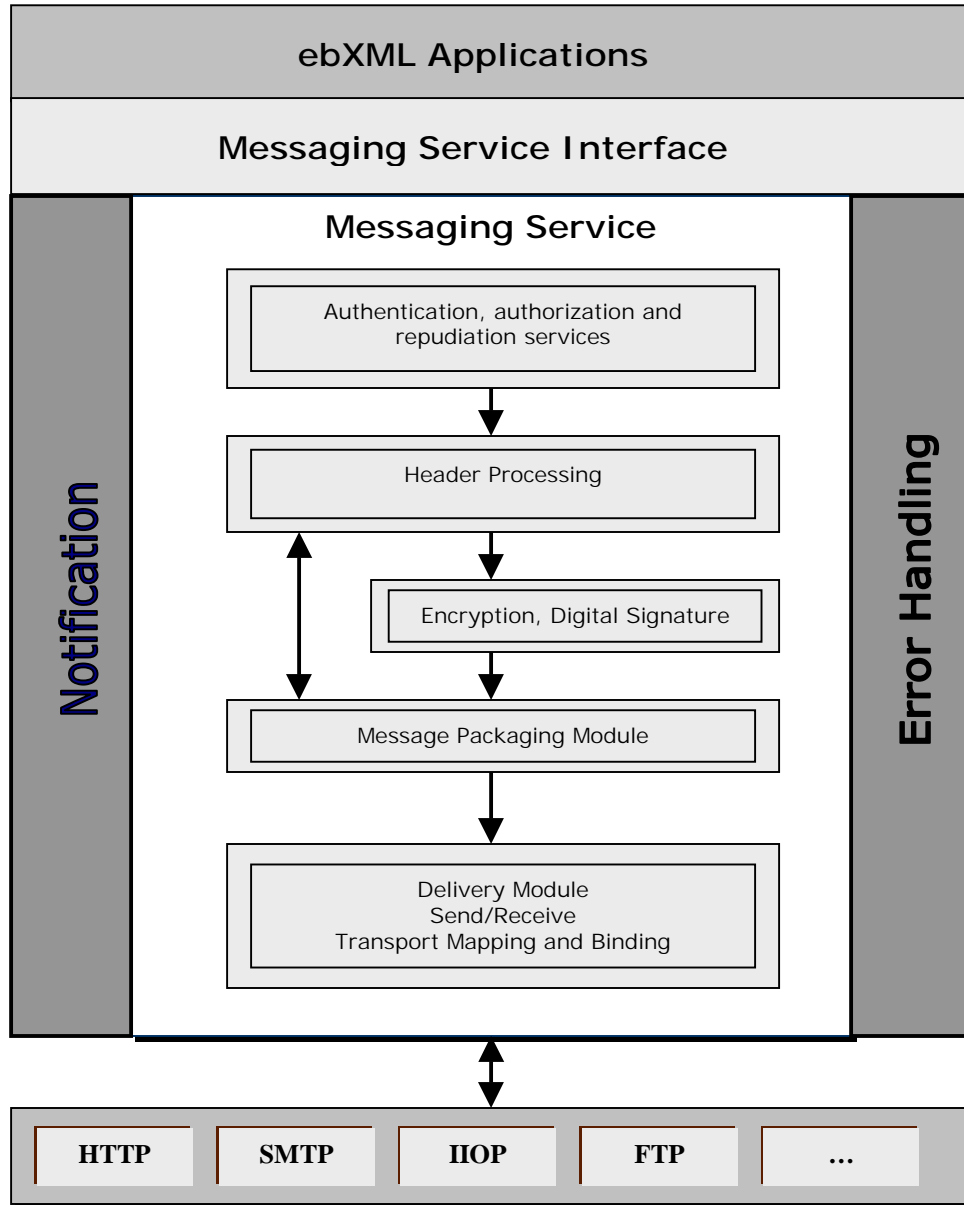1054   Message interchange.
1055

21.3 ebXML Messaging Service Layer Functions

1057   The ebXML Messaging Service Layer provides all of the services and functionality
1058   needed to manage the entire lifecycle of *ebXML Messages*. Functions provided by this
1059   layer include:
1060
1061   •   The ability to construct and validate proper *ebXML Messages.*
1062   •   Enforcing the "rules of engagement" as defined by two parties in a *Trading*
1063   *Partner Agreement* (including security and *Business Process* functions related to
1064   message delivery). The *Trading Partner Agreement* defines the acceptable
1065   behavior by which each *Party* agrees to abide. The definition of these ground

1066	rules can take many forms including formal *Trading Partner Agreements*,
1067	interactive agreements established at the time a business transaction occurs (e.g.
1068	buying a book online), or other forms of agreement. There are Messaging Service
1069	Layer functions that enforce these ground rules. Any violation of the ground rules
1070	result in an error condition, which is reported using the appropriate means.

- 1071 • Support for the following reliability options:
  - 1072 o "Best Effort" delivery
  - 1073 o "Once and only once" delivery
  - 1074 o Synchronous or Asynchronous messaging
  - 1075 o Request/Response processing
  - 1076 o Fire 'n forget processing
  - 1077 o Allow for "multiparty" message delivery
- 1078 • Perform all security related functions including:
  - 1079 o Identification
  - 1080 o Authentication (verification of identity)
  - 1081 o Authorization (access controls)
  - 1082 o Privacy (encryption)
  - 1083 o Integrity (message signing)
  - 1084 o Non-repudiation
  - 1085 o Logging
- 1086 • Interface with internal systems including:
  - 1087 o Routing of received messages to internal systems
  - 1088 o Error notification
- 1089 • Administrative services including:
  - 1090 o Notification, both system-to-system and system-to-human (via pagers or
  - 1091 e-mail)
  - 1092 o Track and report the status of message exchanges for auditing and
  - 1093 diagnostic purposes
  - 1094 o Logging of service related errors
  - 1095 o Access to *Partner Agreement* information
  - 1096 o Status inquiry
- 1097 • Transport bindings:
  - 1098 o Functions to enable the delivery of messages over various transport
 - 1099 services (e.g. SMTP, FTP, HTTP, etc.)
- 1100

1101 The following diagram depicts a logical arrangement of the functional modules that exist
1102 within the ebXML Messaging Service architecture. These modules are arranged in a
1103 manner to indicate their inter-relationships and dependencies. This architecture diagram
1104 illustrates the flexibility of the ebXML Messaging Service, reflecting the broad spectrum
1105 of services and functionality that may be implemented in an ebXML system.
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145

Figure 19: *The Messaging Service Architecture*

1146   **21.4 ebXML Message Structure and Packaging**

1147   Figure 20 below illustrates the logical structure of an ebXML compliant message.
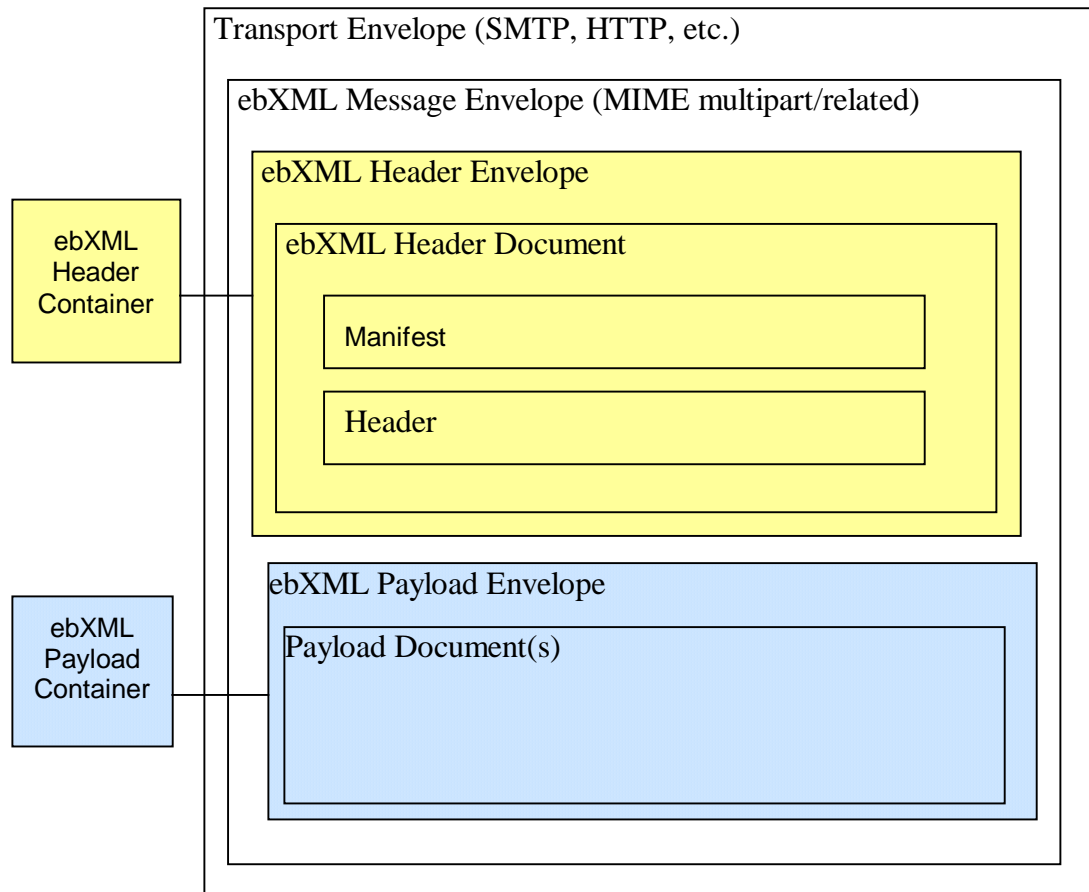
1148

1149   *Figure 20: ebXML Message Structure*

1150

1151   An *ebXML Message* consists of an optional transport protocol specific outer

1152   *Communication Protocol Envelope* and a protocol independent *ebXML Message*

1153   *Envelope.* The *ebXML Message Envelope* is packaged using the MIME multipart/related

1154   content type. MIME is used as a packaging solution because of the diverse nature of

1155   information exchanged between *Partners* in *eBusiness* environments. For example, a

1156   complex B2B business transaction between two or more *Trading Partners* might require

1157   a payload that contains an array of business documents (*XML* or other document

1158   formats), binary images, or other related business objects.

1159

1160   The *ebXML Message Envelope* is used to encapsulate the *Components* of an ebXML

1161   compliant message. This structure effectively separates ebXML header information from

1162   the payload content of the message. The separation of *Header* and *Payload* containers

1163   promotes system efficiency, as the ebXML Messaging Service only needs to access

1164   *Header* information to process the message. This provides a flexible mechanism for

1165   transparently passing diverse *Payloads* to appropriate business services without having to

1166   process them within the Messaging Service framework. It also allows encrypted and/or
1167   signed *Payloads* to be exchanged and forwarded with no processing overhead.
1168
1169   The *ebXML Payload Container* is an optional part of an *ebXML Message*. If a *Payload* is
1170   present in and *ebXML Message*, the *ebXML Payload Envelope* serves as the container for
1171   the actual content (*Payload*) of the *ebXML Message*. The *ebXML Payload Envelope*
1172   consists of a MIME header portion and a content portion (the *Payload* itself). The
1173   ebXML Messaging Service does not limit in any way the structure or content of
1174   payloads.
1175

1176   ## *22.0 Conformance*

1177
1178   ### 22.1 Overview
1179   The objectives of this section are to:
1180   a) Ensure a common understanding of conformance and what is required to claim
1181   conformance;
1182
1183   b) Promote interoperability and open interchange of *Business Processes* and
1184   messages;
1185
1186   c) Promote uniformity in the development of conformance tests.
1187
1188   ebXML conformance is defined in terms of conformance to ebXML, conformance to
1189   each of the component specifications for ebXML, and conformance to this (Technical
1190   Architecture) specification.
1191
1192   All ebXML specifications shall contain a conformance clause. The conformance clause
1193   specifies explicitly all the requirements that have to be satisfied to claim conformance to
1194   that specification. These requirements may be applied and grouped at varying levels
1195   within each specification.
1196
1197   ### 22.2 Conformance Requirements
1198   Types of conformance requirements can be classified as:
1199
1200   a) Mandatory requirements: these are to be observed in all cases;
1201
1202   b) Conditional requirements: these are to be observed if certain conditions set out in
1203   the specification apply;
1204
1205   c) Optional requirements: these can be selected to suit the implementation, provided
1206   that any requirement applicable to the option is observed.
1207
1208   Furthermore, conformance requirements in a specification can be stated:
1209

1210   • Positively: they state what shall be done;
1211   • Negatively (prohibitions): they state what shall not be done.
1212
1213   **22.3 General Framework of Conformance Testing**
1214   The objective of conformance testing is to establish a set of criteria that enable vendors to
1215   implement compatible and interoperable systems built on the ebXML foundations.
1216   Since ebXML consists of many facets and *Components*, ebXML conformance shall take
1217   into account different layers and levels. These levels will be hierarchical and recursive, so
1218   conformance to a higher level will include conformance to a lower level.
1219
1220   Implementations and applications shall be tested to verify their conformance to ebXML
1221   Specifications.
1222
1223   Publicly available test suites from vendor neutral organizations such as OASIS and NIST
1224   should be used to verify the conformance of ebXML implementations, applications, and
1225   *Components* claiming conformance to ebXML. This will ensure that they are compliant
1226   with the base ebXML criteria. Live benchmark implementations may be available to
1227   allow vendors to test their products for interface compatibility and conformance.
1228
1229   Additional items of note include:
1230
1231   a) Extended implementations may include support for more than just the base
1232   ebXML protocols, including other popular or emerging formats, such as legacy
1233   *EDI* or messaging services such as Java Messaging Service (JMS)
1234   implementations.
1235
1236   b) Each ebXML working group will be responsible to coordinate with and determine
1237   what it means to conform to their specification and what should be included in the
1238   appropriate Conformance test suite(s).

1239 ## *Appendix A: Example ebXML Business Scenarios*
1240
1241 ### *Definition*
1242 This set of Scenarios defines how ebXML compliant software could be used to
1243 implement popular, well-known *eBusiness* models.
1244 ### *Scope*
1245 These Scenarios are oriented to properly position ebXML specifications as a convenient
1246 mean for Companies to properly run *eBusiness* over the Internet using open standards.
1247 They bridge the specifications to real life uses.
1248 ### *Audience*
1249 Companies planning to use ebXML compliant software will benefit from these Scenarios
1250 because they will show how these companies may be able to implement popular business
1251 scenarios onto the ebXML specifications.
1252 ### *List*
1253   e)   Two Partners set-up an agreement and run the associated electronic exchange.
1254   f)   Three or more partners set-up a *Business Process* implementing a supply-chain
1255   and run the associated exchanges
1256   g)   A Company sets up a Portal which defines a *Business Process* involving the use
1257   of external business services.
1258   h)   Three or more parties engage in multi-Party *Business Process* and run the
1259   associated exchanges.
1260
1261 ### Scenario 1: Two Partners set-up an agreement and run the associated
1262 ### exchange
1263 In this scenario:
1264 • Each partner defines its own Party Profile.
1265   Each Party Profile references:
1266   • One or more existing *Business Process* found in the ebXML Repository
1267   • One of more Message Definitions. Each Message definition is built from reusable
1268   *Components* (*Core Components*) found in the ebXML Repository
1269
1270   Each Party Profile defines:
1271 • The Commercial Transactions that the Party is able to engage into
1272 • The Technical protocol (like HTPP, SMTP etc) and the technical properties (such as
1273   special encryption, validation, authentication) that the Party supports in the
1274   engagement
1275 • The parties acknowledge each other's profile and create a Partner Agreement.
1276   The Partner Agreement references :
1277 • The relevant Party Profiles
1278 • The Legal terms and conditions related to the exchange
1279 • The parties implement the respective part of the Profile.
1280   This is done:
1281 • Either by creating/configuring a Business Service Interface.
1282 • Or properly upgrading the legacy software running at their side
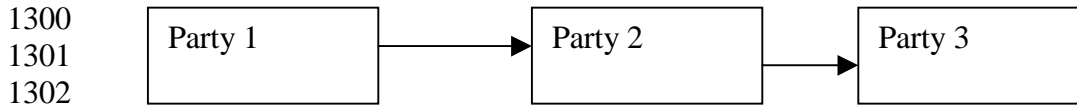1283   In both cases, this step is about :

1284 • Plugging the Legacy into the ebXML technical infrastructure as specified by the
1285   TR&P
1286 • Granting that the software is able to properly engage the stated conversations
1287 • Granting that the exchanges semantically conform to the agreed upon Message
1288   Definitions
1289 • Granting that the exchanges technically conform with the underlying ebXML TR&P
1290 • The parties start exchanging messages and performing the agreed upon commercial
1291   transactions.
1292

1293 **Scenario 2: Three or more partners set-up a Business Process**
1294 **implementing a supply-chain and run the associated exchanges**
1295 The simple case of a supply-chain involving two parties can be reconstructed from the
1296 Scenario 1.
1297 Here we are dealing with situations where more parties are involved. We consider a
1298 *Supply Chain* of the following type :
1299
1300
1301 | Party 1 | → | Party 2 | → | Party 3 |
1302
1303
1304
1305 What fundamentally differs from Scenario 1 is that Party 2 is engaged at the same time
1306 with two different parties. The assumption is that the "state" of the entire *Business*
1307 *Process* is managed by each Party, i.e. that each Party is fully responsible of the
1308 Commercial Transaction involving it (Party 3 only knows about Party 2, Party 2 knows
1309 about Party 3 and Party 1, Party 1 knows about Party 2).
1310
1311 In this scenario:
1312 • Each partner defines its own Party Profile.
1313   Each Party Profile references:
1314 • One or more existing *Business Process* found in the ebXML Repository
1315 • One of more Message Definitions. Each Message definition is built from reusable
1316   *Components* (*Core Components*) found in the ebXML Repository
1317   Each Party Profile defines:
1318 • The Commercial Transactions that the Party is able to engage into
1319   Party 2 must be able to support at least 2 Commercial Transactions.
1320 • The Technical protocol (like HTPP, SMTP etc) and the technical properties (such as
1321   special encryption, validation, authentication) that the Party supports in the
1322   engagement
1323   the technical requirements for the exchanges with Party 1 and Party 3 may be
1324   different. In such case, Party 2 must be able to support different protocols and/or
1325   properties.
1326 • The parties acknowledge each other profile and create the relevant Partner
1327   Agreements (at least 2 in this Scenario).
1328   Each Partner Agreement references:
1329 • The relevant Party Profiles
1330 • The Legal terms and conditions related to the exchange
1331   Party 2 is engaged in 2 Party Agreements.
1332 • The parties implement the respective part of the Profile.
1333   This is done:
1334 • Either by creating/configuring a Business Service Interface.
1335 • Or properly upgrading the legacy software running at their side
1336   In both cases, this step is about:

1337 • Plugging the Legacy into the ebXML technical infrastructure as specified by the
1338   TR&P
1339 • Granting that the software is able to properly engage the stated conversations
1340 • Granting that the exchanges semantically conform to the agreed upon Message
1341   Definitions
1342 • Granting that the exchanges technically conform with the underlying ebXML TR&P
1343 • Party 2 may need to implement a complex Business Service Interface in order to be
1344   able to engage with different partners.
1345 • The parties start exchanging messages and performing the agreed upon commercial
1346   transactions.
1347 • Party 3 places an order at Party 2
1348 • Party 2 (eventually) places an order with Party 1
1349 • Party 1 fulfills the order
1350 • Party 2 fulfill the order
1351

1352 **Scenario 3 : A Company sets up a Portal which defines a Business Process**
1353 **involving the use of external business services**
1354 This is the Scenario describing a Service Provider. A "client" asks the Service Provider
1355 for a Service. The Service Provider fulfills the request by properly managing the
1356 exchanges with other partners, which provide information to build the final answer.
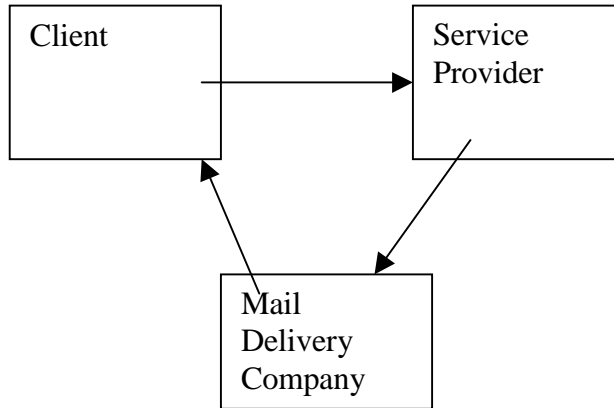1357 In the simplest case, this Scenario could be modeled as follows:
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
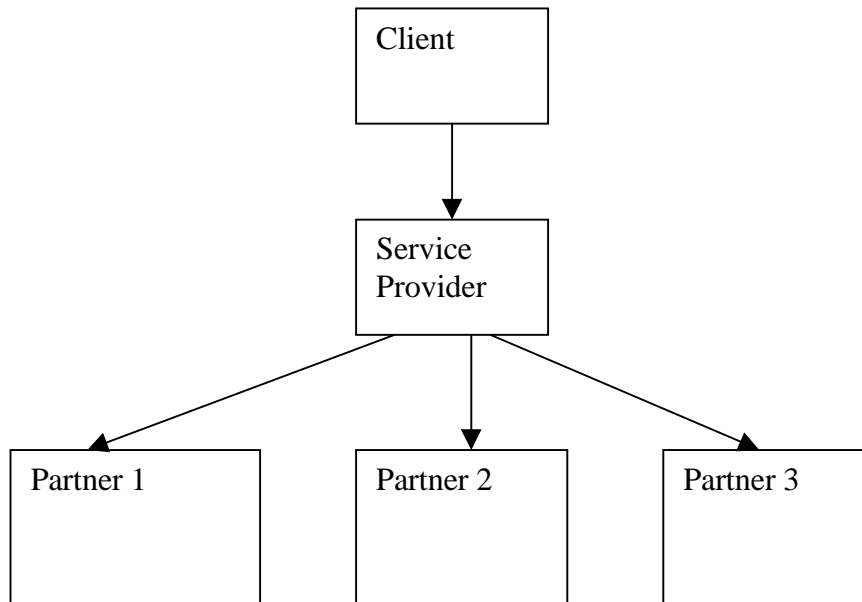
1375 This is an evolution of Scenario 2. The Description of this scenario is omitted.
1376

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

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

```
┌──────────────┐              ┌──────────────┐
│  Client      │────────────▶ │  Service     │
│              │              │  Provider    │
│              │              │              │
└──────────────┘              └──────────────┘
         ▲                       │
          \                     │
           \                   ▼
          ┌──────────────────────┐
          │  Mail                │
          │  Delivery            │
          │  Company             │
          └──────────────────────┘
```

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

In this scenario:

- Each partner defines its own Party Profile.
  Each Party Profile references:
- One or more existing *Business Process* found in the ebXML Repository
- One of more Message Definitions. Each Message definition is built from reusable *Components* (*Core Components*) found in the ebXML Repository
  Each Party Profile defines:
- The Commercial Transactions that the Party is able to engage into
  In this case, each Party must be able to support at least 2 Commercial Transactions.
- The Technical protocol (like HTPP, SMTP etc) and the technical properties (such as special encryption, validation, authentication) that the Party supports in the engagement.
  In case the technical infrastructure underlying the different exchanges differs, each Party must be able to support different protocols and/or properties. (an example is that the order is done through a Web Site and the delivery is under the form of an email).
- The parties acknowledge each other profile and create a Partner Agreement.
  Each Party, in this Scenario, must be able to negotiate at least 2 Agreements.
  The Partner Agreement references :
- The relevant Party Profiles
- The Legal terms and conditions related to the exchange
  Each Party is engaged in 2 Party Agreements.
- The parties implement the respective part of the Profile.
  This is done:
- Either by creating/configuring a Business Service Interface.

1422 • Or properly upgrading the legacy software running at their side
1423   In both cases, this step is about:
1424 • Plugging the Legacy into the ebXML technical infrastructure as specified by the
1425   TR&P
1426 • Granting that the software is able to properly engage the stated conversations
1427 • Granting that the exchanges semantically conform to the agreed upon Message
1428   Definitions
1429 • Granting that the exchanges technically conform with the underlying ebXML TR&P
1430 • All Parties may need to implement complex Business Service Interfaces to
1431   accommodate the differences in the Party Agreements with different Parties.
1432 • The parties start exchanging messages and performing the agreed upon commercial
1433   transactions.
1434 • The Client places an Order at the Service Provider
1435 • The Service Provider Acknowledges the Order with The Client
1436 • The Service Provider informs the Mail Delivery Service about a good to be delivered
1437   at the Client
1438 • The Mail Delivery Service delivers the good at the Client
1439 • The Clients notifies the Service Provider that the good is received.

Disclaimer

1441 The views and specification expressed in this document are those of the authors and are
1442 not necessarily those of their employers. The authors and their employers specifically
1443 disclaim responsibility for any problems arising from correct or incorrect implementation
1444 or use of this design.

Copyright Statement

1446 Copyright © ebXML 2000. All Rights Reserved.

1447

1448 This document and translations of it may be copied and furnished to others, and
1449 derivative works that comment on or otherwise explain it or assist in its implementation
1450 may be prepared, copied, published and distributed, in whole or in part, without
1451 restriction of any kind, provided that the above copyright notice and this paragraph are
1452 included on all such copies and derivative works. However, this document itself may not
1453 be modified in any way, such as by removing the copyright notice or references to the
1454 Internet Society or other Internet organizations, except as needed for the purpose of
1455 developing Internet standards in which case the procedures for copyrights defined in the
1456 Internet Standards process must be followed, or as required to translate it into languages
1457 other than English.

1458

1459 The limited permissions granted above are perpetual and will not be revoked by ebXML
1460 or its successors or assigns.

1461

1462 This document and the information contained herein is provided on an
1463 "AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR
1464 IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE
1465 USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1466 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1467 PARTICULAR PURPOSE.