ebXML Business Process Specification Schema Version 1.05(Candidate for Version 2.0)

Business Process Project Team

15 July 2002

1 Status of this Document

This Technical Specification document is a draft. This material fulfills requirements of the ebXML Requirements document. The formatting for this document is based on the Internet Society's Standard RFC format.

This version:

***Latest version*:**

2 ebXML BP/CoreComponents metamodel participants

Project Team Lead

Brian Hayes (Project Lead), Collaborative Domain

Editors

Pallavi Malu, Intel Corporation: (Project Editor, Issues, XML Schema)

Jean-Jacques Dubray, Eigner: (Editor)

Antoine Lonjon, MEGA International("Transaction Result Computation Diagram")

Ed Buchinski, Treasury Board Secretariat of Canada: (Security)

Arvola Chan, Tibco: (OASIS ebXML Collaboration Protocol Profile and Agreement)

Himagiri (Hima) Mukkamala, Sybase: (OASIS ebXML Messaging, Signal structures)

David Smiley, Mercator: (OASIS ebXML Collaboration Protocol Profile and Agreement)

Participants

Randy Clark, Baker Hughes

Jim Clark, Microsoft

James Bryce Clark, McLure-Moynihan

Karsten Riemer, Sun Microsystems

Larrisa Leybovich, Larrisa Leybovich Consulting

Bob Haugen, Logistical Software

Anders W. Tell, Toolsmiths

Suresh Damodaran, Sterling Commerce

John Yunker, Collaborative Domain

3 ebXML Participants

The authors wish to recognize the following for their significant participation in developing the Business Process Specification Schema, Version 1.01.

Paul Levine, Telcordia

Jim Clark, E2Open - previously Edifecs: (Transaction Semantics)

Cory Casanave, Data Access Technologies: (UML model)

Kurt Kanaskie, Lucent Technologies: (DTD and Examples)

Betty Harvey, Electronic Commerce Connection: (DTD documentation)

Jamie Clark, McLure-Moynihan, Inc.: (Legal aspects)

Neal Smith, Chevron: (Issues Lists, and W3C schema)

John Yunker, Edifecs: (Signal structures)

Karsten Riemer, Sun Microsystems: (Overall Document)

Antoine Lonjon, Mega

J.J. Dubray, Excelon

Bob Haugen, Logistical Software

Bill McCarthy, Michigan State University

Brian Hayes, CommerceOne

Nita Sharma, Netfish

David Welsh, Nordstrom

70 Christopher Ferris, Sun Microsystems
71 Antonio Carrasco, Data Access Technologies
72

4 Table of Contents

5 Introduction

Executive Summary

The ebXML Business Process Specification Schema provides a standard framework by which business systems may be configured to support execution of business collaborations consisting of business transactions. It is based upon prior UN/CEFACT work, specifically the metamodel behind the UN/CEFACT Modeling Methodology (UMM) defined in the N090R10 specification.

The Specification Schema supports the specification of Business Transactions and the choreography of Business Transactions into Business Collaborations. Each Business Transaction can be implemented using one of many available standard patterns. These patterns determine the actual exchange of Business Documents and business signals between the partners to achieve the required electronic commerce transaction.

The current version of the specification schema addresses collaborations between two parties (Binary Collaborations) as well as collaboration involving more than two business partners (Multiparty Collaborations) as a synthesis of binary collaborations.

It is anticipated that a subsequent version will address additional features such as the semantics of economic exchanges and contracts, more complex multi-party choreography, and context based content.

5.1 Summary of Contents of Document

This document describes the ebXML Specification Schema.

This document describes the Specification Schema in its UML form and provides the corresponding XML Schema which every Specification Schema instance must conform to.

The document first introduces general concepts and semantics, then applies these semantics in a detail discussion of each part of the model. The document then specifies all elements in the UML form, and then in XML form.

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

5.2 Audience

The primary audience is business process analysts. We define a business process analyst as someone who interviews business people and as a result documents business processes in unambiguous syntax.

An additional audience is designers of business process definition tools who need to specify the conversion of user input in the tool into the XML representation of the Specification Schema.

The audience is not business application developers.

5.3 Related Documents

As mentioned above, other documents provide detailed definitions of some of the components of the ebXML Business Process Specification Schema and of their inter-relationship. They include ebXML Specifications on the following topics:

- ebXML Technical Architecture Specification, version 1.04
- ebXML Core Components Dictionary, version 1.04
- ebXML Naming Convention for Core Components, version 1.04
- ebXML Collaboration-Protocol Profile and Agreement Specification V2.0
- ebXML Business Process and Business Information Analysis Overview, version 1.0
- ebXML Business Process Analysis Worksheets & Guidelines, version 1.0
- ebXML E-Commerce Patterns, version 1.0
- ebXML Catalog of Common Business Processes, version 1.0
- ebXML Message Service Specification V2.0

- UN/CEFACT Modeling Methodology (UMM) as defined in the N090R10 specification

5.4 Prerequisites

It is assumed that the audience will be familiar with or have knowledge of the following technologies and techniques:

- Business process modeling techniques and principles
- The UML syntax and semantics
- The Extensible Markup Language (XML)

6 Design Objectives

6.1 Goals/Objectives/Requirements/Problem Description

ebXML Business Process Specification Schema models describe interoperable business processes that allow business partners to collaborate. These models must be executed by software components that collaborate on behalf of the business partners.

The goal of the ebXML Business Process Specification Schema is to provide the bridge between e-business process modeling and specification of e-business software components.

The ebXML Business Process Specification Schema provides for the nominal set of specification elements necessary to specify a collaboration between business partners, and to provide configuration parameters for the partners' runtime systems in order to execute that collaboration between a set of e-business software components.

A specification created against the ebXML Business Process Specification Schema is referred to as an ebXML Specification.

The *ebXML Specification Schema* is available as an XML Schema (http://www.w3.org/2001/XMLSchema). This document also provides a UML description of the schema.

The UML version of the *ebXML Business Process Specification Sch*ema is merely a UML Class Diagram. It is not intended for the direct creation of ebXML Business Process Specifications. Rather, it is a self-contained statement of all the specification elements and relationships required to be able to create an ebXML compliant Business Process Specification. Any methodologies and/or metamodels used for the creation of ebXML compliant Business Process Specifications must at minimum support these elements and relationships.

The XML Schema provides the specification for XML based instances of ebXML Specifications.

The UML and XML based representations of the *ebXML Business Process Specification Schema* are unambiguously mapped to each other.

6.2 Caveats and Assumptions

This specification is designed to specify the run time aspects of a business collaboration.

It is not intended to incorporate a methodology, and does not directly prescribe the use of a methodology. However, if a methodology is to be used, it is recommended that it be UN/CEFACT Modeling Methodology (UMM).

The *ebXML Business Process Specification Schema* does not by itself define Business Documents Structures. It is intended to work in conjunction with already existing Business Document definitions, and/or the document metamodel defined by the ebXML Core Components specifications.

6.2.1 Relationship between *ebXML Business Process Specification Schema* and UMM

The UN/CEFACT Modeling Methodology (UMM) is a methodology for business process and information modeling.

This section describes the relationship between UMM and the ebXML *Business Process Specification Schema*.

The UMM Meta Model is a description of business semantics that allows Trading Partners to capture the details for a specific business scenario (a Business Process) using a consistent modeling methodology. A Business Process describes in detail how Trading Partners take on shared roles, relationships and responsibilities to facilitate interaction with other Trading Partners. The interaction between roles takes place as a choreographed set of Business Transactions. Each Business Transaction is expressed as an exchange of electronic Business Documents. The sequence of the exchange is determined by the Business Process, and by messaging and security considerations. Business Documents are composed from re-useable Business Information Objects. At a lower level, Business Processes can be composed of re-useable Common Business Processes, and Business Information Objects can be composed of re-useable Core Components. Common Business Processes and Business Information Objects reside in a UMM Business Library.

The UMM Meta Model supports a set of Business Process viewpoints that provide a set of semantics (vocabulary) for each viewpoint and forms the basis for specification of the semantics and artifacts that are required to facilitate business process and information integration and interoperability. Using the UMM methodology and the UMM metamodel, the user may thus create a complete Business Process and Information Model. This model contains more information than what is required for configuring ebXML compliant software. Also the model is syntax independent and not directly interpretable by ebXML compliant software.

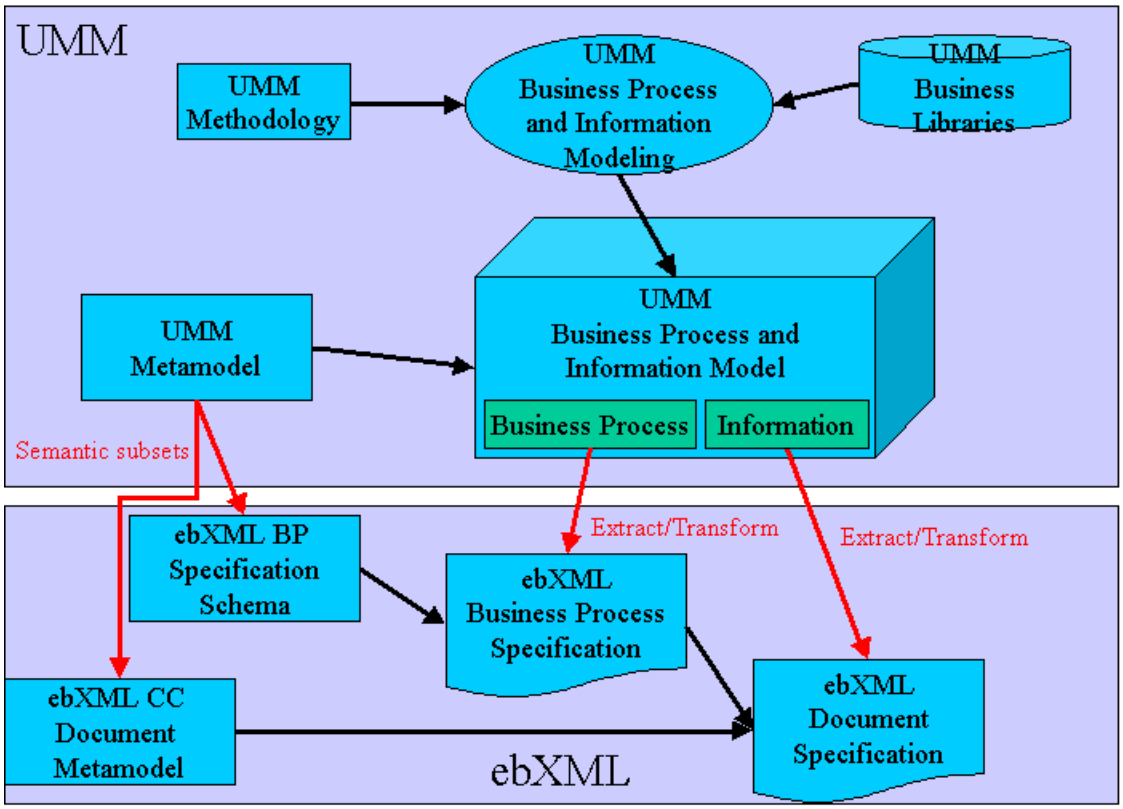
The *ebXML Business Process Specification Schema* provides an additional view of the UMM metamodel. This subset is provided to support the direct specification of the nominal set of elements necessary to configure a runtime system in order to execute a set of ebXML business transactions. By drawing out modeling elements from several of the other views, the *ebXML Business Process Specification Schema* forms a semantic subset of the UMM Meta Model. Using the *ebXML Business Process Specification Schema* the user may thus create a Business Process Specification that contains only the information required to configure ebXML compliant software.

It is expected that ebXML compliant software will be configured with XML instances conforming to the ebXML Business Process Specification Schema.

The relationship between the UMM Meta Model and the *ebXML Business Process Specification Schema* is shown in Figure 1.

311 **Figure 1. UMM Metamodel and *ebXML Business Process Specification Schema***

312



313

314 Using the UMM methodology, and drawing on content from the UMM Business
315 Library a user may create complete Business Process and Information Model
316 conforming to the UMM metamodel.

317 Since the ebXML *Business Process Specification Schema* is a semantic subset
318 of the UMM metamodel, the user may then in an automated fashion extract from
319 the Business Process and Information Model the required set of elements and
320 relationships, and transform them into an ebXML Business Process Specification
321 conforming to the *ebXML Business Process Specification Schema*.

322 Likewise, since the ebXML CC document metamodel is aligned with the UMM
323 Metamodel, the user may then in an automated fashion extract from the
324 Business Process and Information Model the required set of elements and
325 relationships, and transform them into an ebXML document model conforming to
326 ebXML Core Component specifications.

327 The UMM methodology is not part of the formal set of ebXML specifications.

328 Likewise, the UMM metamodel in its entirety is not part of the formal set of
329 ebXML specifications. Only the semantic subset represented by the *ebXML*
330 *Business Process Specification Schema* and CC are parts of the formal set of
331 ebXML specifications.

The remainder of this document focuses on the *ebXML Business Process Specification Schema* and Business Process Specifications created against it. It is understood that proper Business Process and Information Modeling may have taken place prior to beginning the activity of creating a Business Process Specification.

7 System Overview

The ebXML *Business Process Specification Schema* provides a standard framework for business process specification. As such, it works with the ebXML Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA) specifications to bridge the gap between Business Process Modeling and the configuration of ebXML compliant e-commerce software, e.g. an ebXML Business Service Interface, as depicted in Figure 2.

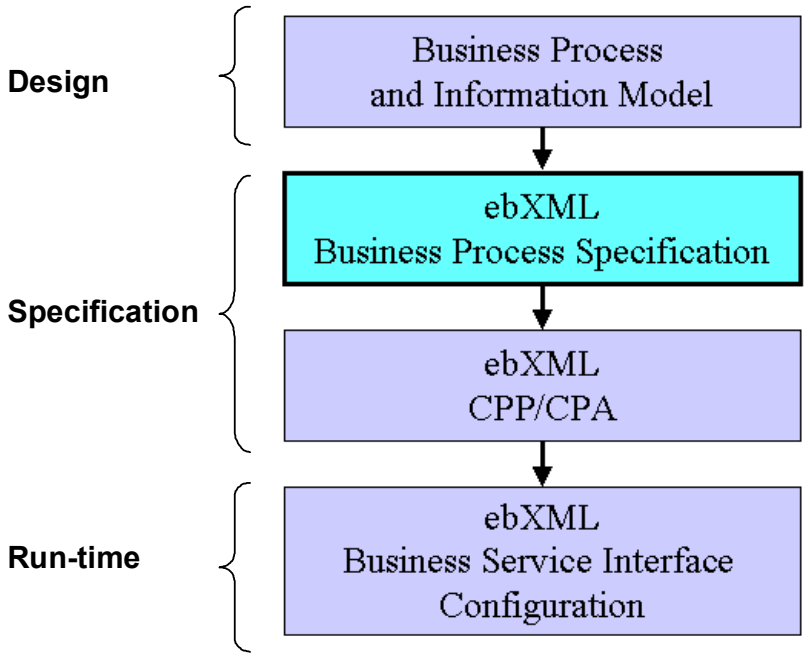


Figure 2: Business Process Specification and Business Service Interface Configuration

Using Business Process Modeling, a user may create a complete Business Process and Information Model.

Based on this Business Process and Information Model and using the ebXML *Business Process Specification Schema* the user will then extract and format the nominal set of elements necessary to configure an ebXML runtime system in

order to execute a set of ebXML business transactions. The result is an ebXML *Business Process Specification*.

Alternatively the ebXML *Business Process Specification* may be created directly, without prior explicit business process modeling.

An ebXML *Business Process Specification* contains the specification of Business Transactions and the choreography of Business Transactions into Business Collaborations.

This ebXML *Business Process Specification* is then the input to the formation of ebXML trading partner Collaboration Protocol Profiles and Collaboration Protocol Agreements.

These ebXML trading partner Collaboration Protocol Profiles and Collaboration Protocol Agreements in turn serve as configuration files for ebXML Business Service Interface software. The ebXML Business Interface Software represents any ebXML compliant component, which is able to be, configured from an ebXML Specification Schema, a CPP and a CPA.

The architecture of the ebXML *Business Process Specification Schema* consists of the following functional components:

- UML version of the *Business Process Specification Schema*
- XML version of the *Business Process Specification Schema*
- Production Rules defining the mapping from the UML version of the *Business Process Specification Schema* to the XML version
- Business Signal Definitions

Together these components allow you to fully specify all the run time aspects of a business process model.

These components are shown (inside the dotted box) in figure 3 below.

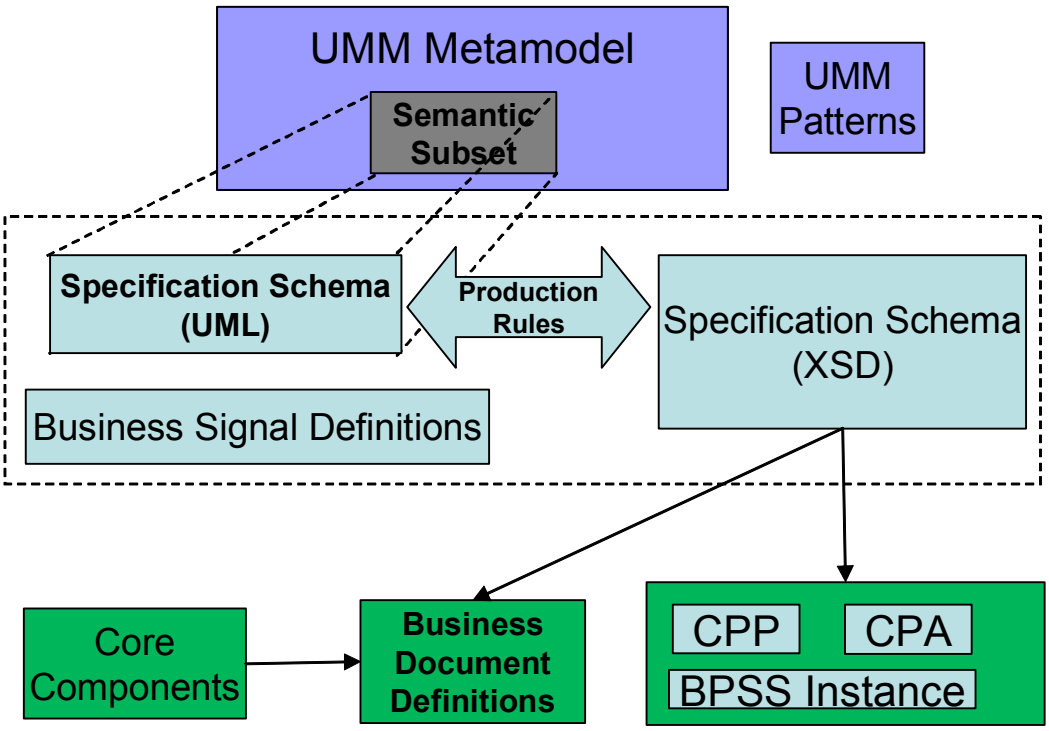


Figure 3: Relationship of *ebXML Business Process Specification Schema* to UMM, CPP/CPA and Core Components

The following provides a description of each of the components in the ebXML *Business Process Specification Schema* and their relationship to UMM, and ebXML CC and CPP/CPA:

UML Representation of Business Process Specification Schema

The UML representation of the *ebXML Business Process Specification Schema* is a semantic subset of the metamodel behind UMM as specified in UN/CEFACT TMWG's N090R10

N090R10 is as of this writing not yet approved by UN/CEFACT. It is the intent to keep the *ebXML Business Process Specification Schema and the* UN/CEFACT TMWG's N090 semantically aligned.

The UML version of the ebXML *Business Process Specification Schema* is merely a UML Class Diagram. It is not intended for the direct creation of ebXML Business Process Specifications. Rather, it is a self-contained statement of all the specification elements and relationships required to be able to create an ebXML compliant Business Process Specification.

XML Schema representation of Business Process Specification Schema

The corresponding XML Schema representation of the ebXML *Business Process Specification Schema* provides the specification for XML based instances of ebXML Business Process Specifications, and as a target for production rules from other representations. Thus, a user may either create a *Business Process Specification* directly as an XML document, or may chose to use some other means of specification first and then apply production rules to arrive at the XML document version.

Any methodologies and/or metamodels used for the creation of ebXML compliant Business Process Specifications must at minimum support the production of the elements and relationships contained in the XML representation of the ebXML *Business Process Specification Schema.*

This XML Schema definition is isomorphic to the UML representation of the ebXML *Business Process Specification Schema*.

UMM Business Process Interaction Patterns

Any ebXML Business Service Interface software components should be able to be configured to execute the business processes specified in a *Business Process Specification*. They do so by exchanging ebXML messages and business signals.

Each Business Transaction can be implemented using one of many available standard patterns. These patterns determine the actual exchange of messages and business signals between the partners to achieve the required electronic commerce transaction.

The Business Transaction Interaction Patterns set forth in Chapter 8 of the UMM N090R10 document illustrate recommended permutations of message sequences as determined by the type of business transaction defined and the timing policies specified in the transactions.

While the UMM patterns themselves are not part of the ebXML specifications, all the security and timing parameters required to express the pattern properties are provided as attributes of elements in the ebXML *Business Process Specification Schema*.

Business Signal Definitions

Business signals are application level documents that ‘signal’ the current state of the business transaction. These business signals have specific business purpose and are separate from lower protocol and transport signals as specified in the ebXML Message Service Specification.

However, the structures of ebXML business signals are ‘universal’ and do not vary from transaction to transaction. Thus, they can be defined once and for all as part of the ebXML *Business Process Specification* itself.

The Business Process Specification provides both the choreography of business signals, and the structure definition of the business payload of a business signal. The ebXML Message Service Specification signal structures provide business service state alignment infrastructure, including unique message identifiers and digests used to meet the basic process alignment requirements. The business signal payload structures provided herein are optional and normative and are intended to provide business and legal semantics to the business signals.

A Schema is provided for each of the possible business signals.

Production Rules

A set of production rules is provided, defining the mapping from the UML version of the ebXML *Business Process Specification Schema* to the XML version.

The primary purpose for these production rules is to govern the one-time generation of the Schema representation of the ebXML *Business Process Specification Schema* from the UML Class Diagram version of the ebXML *Business Process Specification Schema*.

The Class Diagram representation of *Business Process Specification Schema* is not intended for the direct creation of ebXML Business Process Specifications. However, if a *Business Process Specification* were in fact (programmatically) created as an instance of this class diagram, the production rules would also apply for its conversion into a Schema conformant XML document.

Separately, it is expected that a set of production rules will be constructed for the production of an XML version of an ebXML *Business Process Specification* from a set of UML diagrams constructed through the use of UMM.

An instance of the UML Class Diagram version of the ebXML *Business Process Specification Schema* will through the application of its production rules produce an XML Specification Document that is analytically, semantically and functionally equivalent to one arrived at by modeling the same subset through the use of UMM and its associated production rules.

Relationship to CPP/CPA

A *Business Process Specification* is, along with protocol specifications, the object of the agreement between two parties. The *Business Process Specification* is therefore incorporated with or referenced by ebXML trading partner Collaboration Protocol Profiles (CPP) and Collaboration Protocol Agreements (CPA). Each CPP declares its support for one or more Roles within the *Business Process Specification*. A Business Process Specification is also a machine interpretable specification needed for an ebXML Business Service Interface, which will enforce its definition at run-time. The CPP profiles and CPA agreements contain further technical parameters resulting in a full specification of the run-time software at each trading partner.

Relationship to CC

The *Business Process Specification Schema* does not by itself support the definition of Business Documents. Rather, a *Business Process Specification* merely points to the definition of Business Documents. Such definitions may either be XML based, or – as attachments – may be any other structure, or completely unstructured. XML based Business Document Specifications may be based on the ebXML Core Components specifications.

Relationship to ebXML Message Service Specification

The Business Process Specification Schema will provide choreography of business messages and signals. The ebXML Message Service Specification provides the infrastructure for message / signal identification, typing, and integrity; as well as placing any one message in sequence with respect to other messages in the choreography.

7.1 Key Concepts of the ebXML Business Process Specification Schema

The ebXML *Business Process Specification Schema* provides the semantics, elements, and properties necessary to define business collaborations.

A business collaboration consists of a set of roles collaborating through a set of choreographed transactions by exchanging business documents.

These basic semantics of a business collaboration are shown in Figure 4.

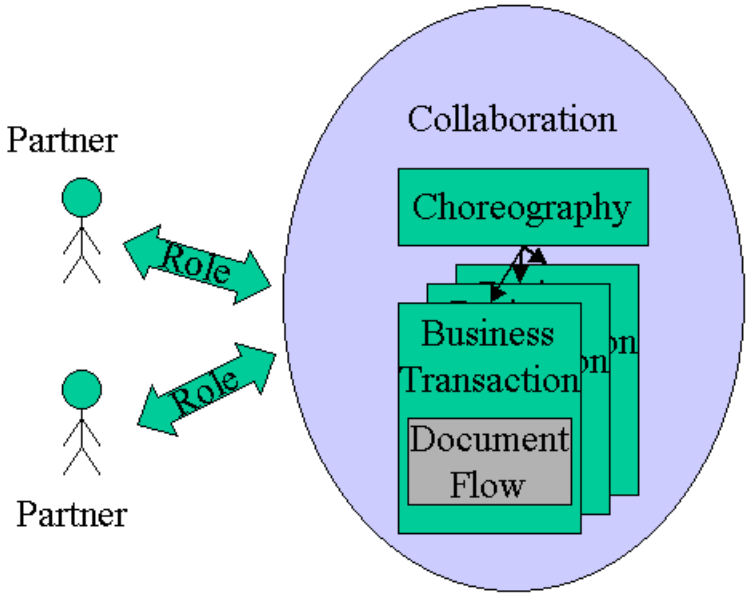


Figure 4. Basic Semantics of a business collaboration

Two or more business partners participate in the business collaboration through roles. The roles interact with each other through Business Transactions. The business transactions are sequenced relative to each other in Choreography. Each Business Transaction consists of one or two predefined Business document flows. One or more Business Signals may additionally support a Business Transaction.

The following section describes the concepts of a Business Collaboration, a Business Transaction, a Business document flow, and Choreography

1. Business Collaborations

   A business collaboration is a set of Business Transactions between business partners. Each partner plays one or more roles in the collaboration.

   The ebXML *Business Process Specification Schema* supports two levels of business collaborations, Binary Collaborations and Multiparty Collaborations.

   Binary Collaborations are between two roles only.

Multiparty Collaborations are between more than two roles, but such Multiparty Collaborations are always synthesized from two or more Binary Collaborations. For instance if Roles A, B, and C collaborate and all parties interact with each other, there will be a separate Binary Collaboration between A and B, one between B and C, and one between A and C. The Multiparty Collaboration will be the synthesis of these three Binary Collaborations.

Binary Collaborations are expressed as a set of Business Activities between the two roles. Each Business Activity reflects a state in the collaboration. The Business Activity can be a Business Transaction Activity, i.e. the activity of conducting a single Business Transaction, or a Collaboration Activity, i.e. the activity of conducting another Binary Collaboration. An example of the former is the activity of placing a purchase order. An example of the latter is the activity of negotiating a contract. In either case the activities can be choreographed relative to other activities as per below.

The ability of a Binary Collaboration to have activities that in effect are executing other Binary Collaborations is the key to recursive compositions of Binary Collaboration, and to the re-use of Binary Collaborations. An activity, whether it is a Business Transaction Activity or a Collaboration Activity represents the usage of a definition within a Binary Collaboration Specification. For instance, a Business Transaction is defined once and for all, but could appear several times – as a Business Transaction Activity -, sometimes even with opposite roles, within the same binary collaboration definition.

In essence each Binary Collaboration is a re-useable protocol between two roles.

2. Business Transactions

A Business Transaction is the atomic unit of work in a trading arrangement between two business partners. A Business Transaction is conducted between two parties playing opposite roles in the transaction. The roles are always a requesting role and a responding role. They are not specific roles like buyer or seller. These roles will be specified at the Business Transaction Activity level, when the Business Transaction definition is used for a specific purpose.

Like a Binary Collaboration, a Business Transaction is a re-useable protocol between two roles. The way it is re-used is by referencing it from a Binary Collaboration through the use of a Business Transaction Activity as per above. In a Business Transaction Activity the roles of the Binary Collaboration are assigned to the execution of the Business Transaction.

Unlike a Binary Collaboration, however, the Business Transaction is atomic; it cannot be decomposed into lower level Business Transactions.

A Business Transaction is a very specialized and very constrained protocol, in order to achieve very precise and enforceable transaction semantics. These semantics are expected to be enforced by the software

564 managing the transaction, i.e. an ebXML Business Service Interface (BSI)
565 software component.

566 A Business Transaction will always either succeed or fail both from a
567 protocol and a business perspective. If it succeeds from both perspective
568 it may be designated as legally binding between the two partners, or
569 otherwise govern their collaborative activity. If it fails it is null and void,
570 and each partner must relinquish any mutual claim established by the
571 transaction. In addition, if it fails from protocol perspective, each party
572 must synchronize their state to the state prior the start of the transaction.
573 In case of a business failure, the state has already been synchronized
574 and it is the duty of each application to take the proper actions.

575 3. Business Document flows

576 A business transaction is realized as Business Document flows between
577 the requesting and responding roles. There is always a requesting
578 Business Document, and optionally a responding Business Document,
579 depending on the desired transaction semantics, e.g. one-way notification
580 vs. two-way conversation.

581 Actual document definition is achieved using the ebXML core component
582 specifications, or by some methodology external to ebXML but resulting in
583 Schema definition (XSD or DTD) that an ebXML *Business Process*
584 *Specification* can point to.

585 4. Choreography

586 The Business Transaction Choreography describes the ordering and
587 transitions between business transactions or sub collaborations within a
588 binary collaboration. For example, in a UML tool this could be done using
589 a UML activity diagram. The choreography is described in the ebXML
590 *Business Process Specification Schema* using activity diagram concepts
591 such as start state, completion state, activities, synchronizations,
592 transitions between activities, and guards on the transitions. This
593 document does not specify a graphical representation of collaboration.

594 5. Patterns

595 The ebXML *Business Process Specification Schema* provides a set of
596 unambiguous semantics within which to specify transactions and
597 collaborations. Within these semantics the user community has flexibility
598 to specify an infinite number of specific transactions and collaborations.
599 The use of predefined patterns combines this flexibility with a consistency
600 that facilitates faster design, faster implementation, and enables generic
601 processing.

602 A set of predefined transaction interaction patterns, defining common
603 combinations of transaction interaction parameter settings can be found
604 in UMM.

605 While the UMM transaction interaction patterns themselves are not part of
606 the ebXML specifications, all the security and timing parameters required

607 to express the pattern properties are provided as attributes of elements in
608 the *Business Process Specification Schema*.

609 It is also anticipated that patterns for collaboration choreographies will
610 emerge. An example of such a pattern is in the ebXML E-Commerce
611 Patterns.

612 Re-use, recursion, and patterns are among the key concepts of the ebXML
613 *Business Process Specification Schema*. The following section will illustrate
614 these key concepts.

615 *7.2 How to use the ebXML Business Process Specification*
616 *Schema*

617 The ebXML *Business Process Specification Schema* should be used wherever
618 ebXML compliant software is being specified to execute Business Collaborations.
619 The generic term for such software is a Business Service Interface (BSI).

620 The ebXML *Business Process Specification Schema* is used to specify the
621 business process related configuration parameters for configuring a BSI to
622 execute these collaborations.

623 This section discusses

624 • How the ebXML *Business Process Specification Schema* fits in with other
625 ebXML specifications.

626 • How to use the ebXML *Business Process Specification Schema* at design
627 time, either for specifying brand new collaborations and transactions, or
628 for re-using existing ones.

629 • How to specify core transaction semantics and parameters needed for a
630 Collaboration-Protocol Profile and Agreement (CPP/CPA).

631 • Run-time transaction and collaboration semantics that the ebXML
632 *Business Process Specification Schema* specifies and the Business
633 Service Interface (BSI) is expected to manage.

634 7.3 How ebXML *Business Process Specification Schema* is
635 used with other ebXML specifications

636

637 The ebXML *Business Process Specification Schema* provides the semantics,
638 elements, and properties necessary to define Business Collaborations.

639 A collaboration consists of a set of roles collaborating through a set of
640 choreographed transactions by exchanging Business Documents.

641 As shown in Figure 5, Business Documents are defined at the intersection
642 between the Business Process Specification and the ebXML Core Component
643 specifications. A Business Process Specification will reference, but not define, a
644 set of required Business Documents. At ebXML Business Documents are either
645 defined by some external document specification, or assembled directly or
646 indirectly from lower level information structures called core components. The

assembly is based on a set of contexts, many of which are provided by the business processes, i.e. collaborations that use the documents in their document flows.

The combination of the business process specification and the document specification become the basis against which partners can make agreements on conducting electronic business with each other.

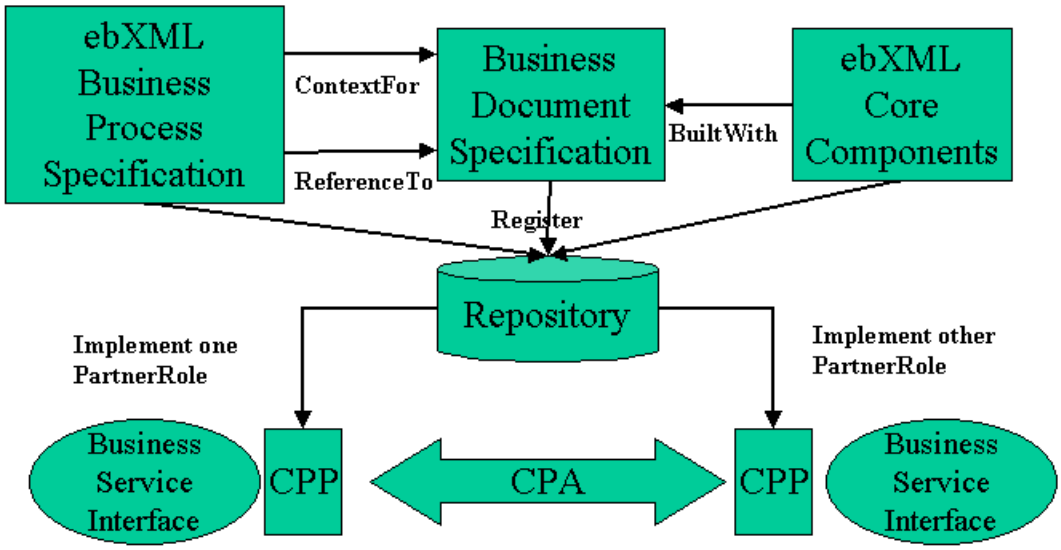


Figure 5: ebXML *Business Process Specification Schema* and other ebXML Specifications

The user will extract and transform the necessary information from an existing Business Process and Information Model. Associated production rules could aid in creating an XML representation of a *Business Process Specification.*

Alternatively a user would use an XML based tool to produce the XML representation directly. Production rules could then aid in converting into XMI, so that it could be loaded into a UML tool, if required.

In either case, the XML representation of the *Business Process Specification* gets stored in the ebXML repository and registered in the ebXML registry for future retrieval. The Business *Process Specification would* be registered using classifiers derived during its design.

When implementers want to establish trading partner Collaboration Protocol Profile and Agreement the *Business Process Specification* XML document, or the relevant parts of it, are simply imbedded in or referenced by the CPP and CPA XML documents. ebXML CPP and CPA documents can only reference ebXML *Business Process Specifications* and only XML versions thereof.

Guided by the CPP and CPA specifications the resulting XML document then becomes the configuration file for one or more Business Service Interfaces (BSI), i.e. the software that will actually manage either partner's participation in the collaboration.

7.4 How to design collaborations and transactions, re-using at design time

This section describes the ebXML *Business Process Specification Schema* modeling relationships by building a complete Multiparty Collaboration from the bottom up, as follows:

1. Specify a Business Transaction
2. Specify the Business Document flow for a Business Transaction
3. Specify a Binary Collaboration re-using the Business Transaction
4. Specify a Choreography for the Binary Collaboration
5. Specify a higher level Binary Collaboration re-using the lower level Binary Collaboration
6. Specify a Multiparty Collaboration re-using Binary Collaborations

Although this section, for purposes of introduction, discusses the specification of collaboration from the bottom up, the ebXML *Business Process Specification Schema* very much is intended for specifying collaborations from the top down, re-using existing lower level content as much as possible.

The constructs listed above support the specification of fairly complex multi party collaborations. However, an ebXML compliant Business Process Specification may be as simple as a single Binary Collaboration referencing a single Business Transaction. This involves only numbers 1 through 3 above. In other words, Higher-level Binary Collaborations, Multi-party Collaborations and choreography expressions are not required for ebXML Business Process Specification compliance.

7.4.1 Packages and Includes

All model elements of this specification are defined within the context of a package. Packages may contain other package, therefore defining a hierarchy of packages.

A package defines the namespace of the elements inside it. You cannot have two model elements with the same name within the same package. Model element names can be qualified with the package using the Java notation:

Business Transaction.Order Entry.Process Purchase Order

Which means that the Process Purchase Order business transaction is defined within the package Order Entry, which is itself, defined within the Business Transaction package. Note that there is no ambiguity in using spaces within the names of packages or model elements.

If a model element in package Order Entry needs to name something in a package called Billing, it must include this package to make its elements visible to its own model elements. Unlike an import, include requires that all model elements from the Billing package be fully qualified. So if we want to designate the Invoice business document within the Order Entry.Process Purchase Order transaction we need to refer to the Billing.Invoice document, assuming it is defined in the Business Transaction.Billing package.

7.4.2 Substitution Sets

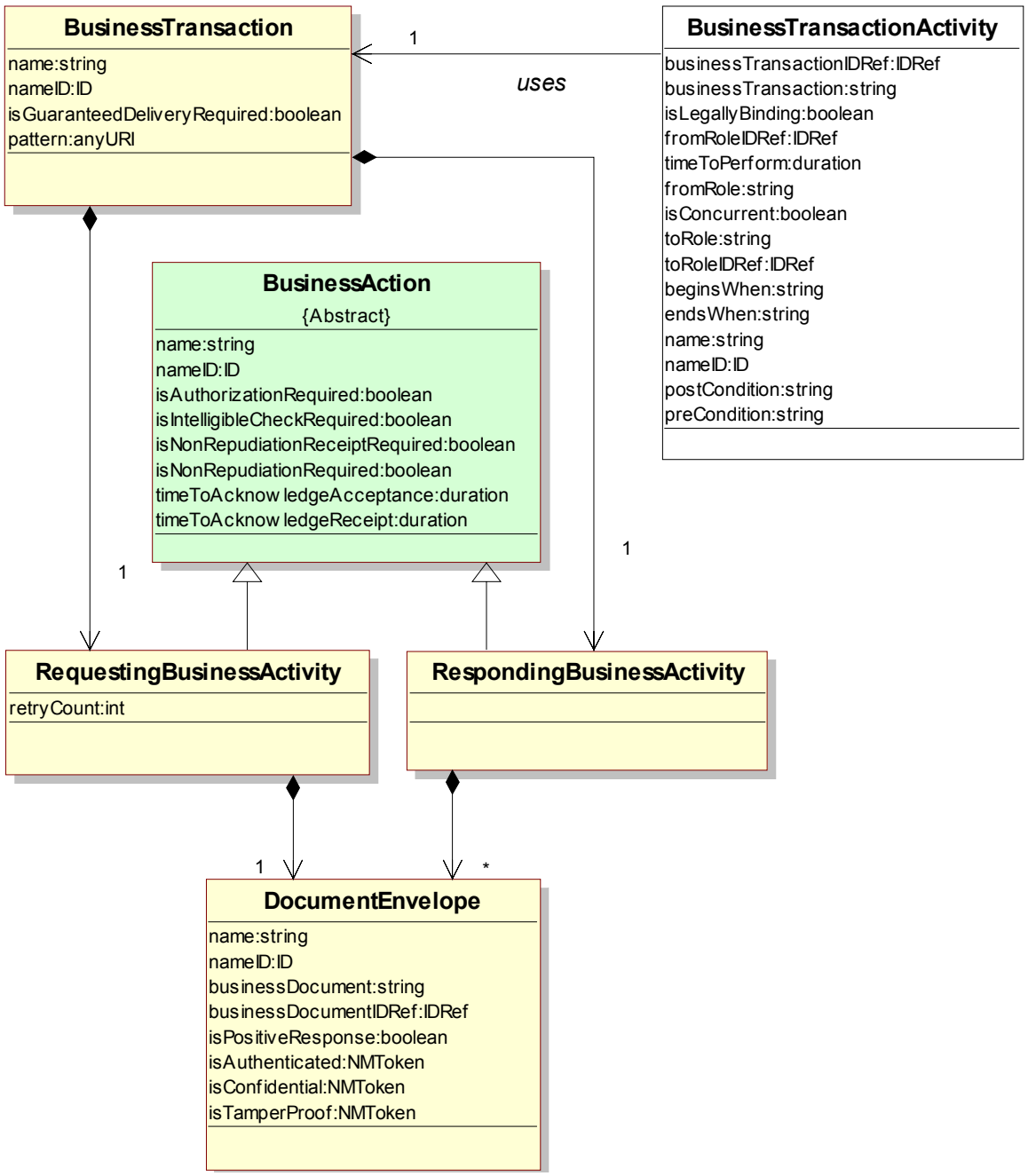
There is a requirement for Business specifications that are less coupled to technology and business details, such as specific document formats and structures and timing parameters. Substitution sets support the capability to take a generic business process and specialize it for a specific use. For example, an ordering process may be very generic but a specific use of that process may require specific document capabilities that go beyond the generic.

A substitution set is placed in the more specific process specification and replaces or makes more explicit document definition references and attribute values. A Substitution Set is a container for one or more AttributeSubstitution and/or DocumentSubstitution elements. The entire SubstitutionSet specifies document or attribute values that should be used in place of some documents and attribute values in an existing process specification.

7.4.3 Specify a Business Transaction and its Business Document Flow

Figure 6 illustrates a business transaction.

739



740

741 **Figure 6. UML Diagram of a Business Transaction**

742

743 ### 7.4.3.1 Key Semantics of a Business Transaction

744

745 A Business Transaction is the atomic unit of work in a trading
746 arrangement between two business partners.

A business transaction consists of a Requesting Business Activity, a Responding Business Activity, and one or two document flows between them. One or more Business Signals that govern the use and meaning of acknowledgements and related matters in the transaction may additionally support a Business Transaction.

Implicitly there is a requesting role performing the Requesting Business Activity and a responding role performing the Responding Business Activity. These roles become explicit when the transaction is used within a Business Transaction Activity within a Binary Collaboration. There is no need to make these roles more explicit such as buyer or seller. In particular some business transactions, for example “Cancel Purchase Order” may be used either way within the same binary collaboration definition.

There is always a Request document flow.

Whether a Response document is required is part of the definition of the Business Transaction. Some Business Transactions need this type of request and response, typically for the formation of a contract or agreement. Other Business Transactions are more like notifications, and have only a Request document flow.

An abstract superclass, Business Action, is the holder of attributes that are common to both Requesting Business Activity and Responding Business Activity.

There is a retryCount attribute on the Requesting Business Activity, which indicate the number of business level retries, in case there is a business protocol failure. The message id is different when retried and the receiving application will be responsible for detecting for business level duplicates using the unique id (like a purchase order number) in the business information.

7.4.3.2 Sample syntax

Here is a simple business transaction definition with just a requesting and responding document flow:

```
<BusinessTransaction name="Get Repair Order">
      <RequestingBusinessActivity name="Dealer"
         <DocumentEnvelope
          businessDocument="Get Repair Order"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name="OEM">
          <DocumentEnvelope isPositiveResponse="false"
            businessDocument="confirmBOD"/>
          <DocumentEnvelope isPositiveResponse="true"
            businessDocument="Show Repair Order" />
      </RespondingBusinessActivity>
 </BusinessTransaction>
```

Associated with each document flow can be one or more business signals acknowledging the document flow. These acknowledgment signals are not modeled explicitly but parameters associated with the transaction specify whether the signals are required or not.

The possible Document Flows and business signals within a Business Transaction are shown in Figure 7.

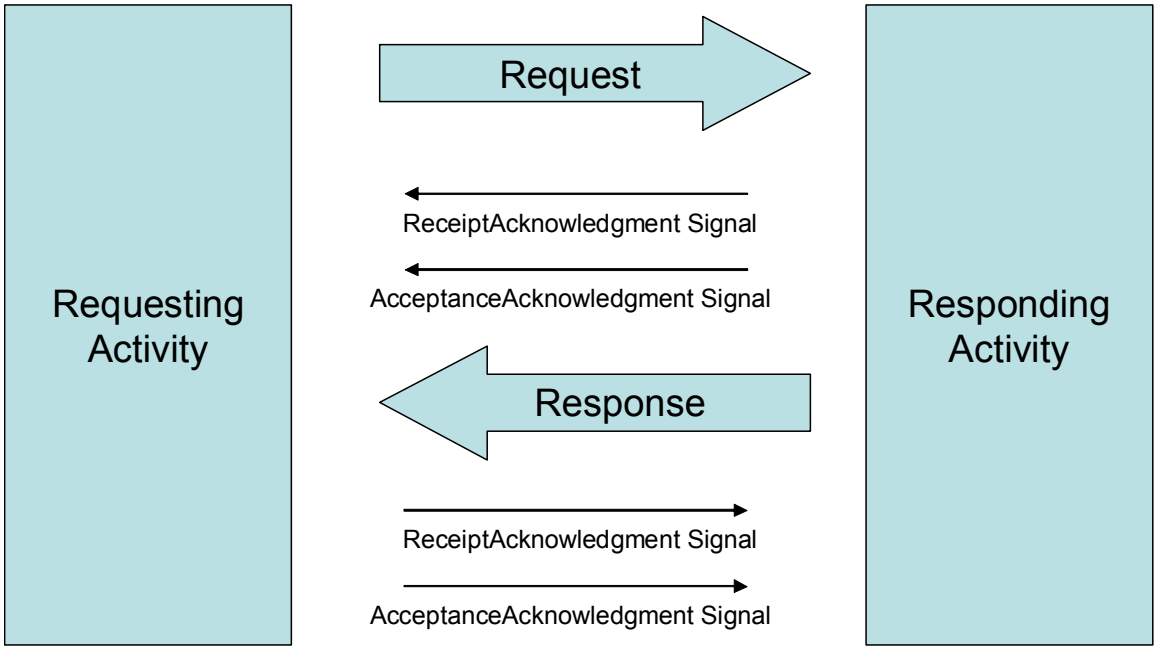


Figure 7: **Possible document flows and signals and their sequence**

These acknowledgment signals (a.k.a. Business Signals) are application level documents that 'signal' the current state of the business transaction.

Whether a receiptAcknowledgement and/or acceptanceAcknowledgement signal is required is part of the pattern specified for the Business Transaction. These business signals have specific business purposes, relating to the processing and management of documents and document envelopes *prior* to evaluation of their business terms, and are separate from lower protocol and transport signals.

The Receipt acknowledgement business signal, if used, signals that a message has been properly received by the ebXML Business Service Interface software component. The property ***isIntelligibleCheckRequired*** allows partners to agree that a message should be confirmed by a Receipt acknowledgement only if it is also legible. Legible means that it has passed structure/ schema validity check. Both the proper receipt and, if evaluated, the legibility of a message are reviewed (and if present acknowledged) *prior* to the application of any business rules or evaluation

of the terms or guard expressions in the message's business documents or document envelope.

The Acceptance Acknowledgement business signal, if used, signals that the message received has been accepted for business processing by the receiving application, or a receiving business application proxy. This is the case if the contents of the message's business documents and document envelope have passed a business rule validity check. These business rules are not necessarily modeled as part of the collaboration.

Failure to send either signal, when required (by specifying a timeout value in timeToAcknowledgeReceipt or timeToAcknowledgeAcceptance), will result in the transaction being null and void, and therefore will prevent any "success" end state (protocol or business) that would have depended on receipt of a business document satisfying the associated timeToPerform.

Conversely, if all signals are positive and sent and received on time, the transaction will be successful from a protocol perspective.

The isPositiveResponse attribute of a DocumentEnvelope is not part of the business transaction protocol and therefore does not impact the protocol success or failure of a collaboration. It the DocumentEnvelope received as a response is specified with the isPositiveResponse=false (at design time) the business transaction will end in a business failure state. The choreography of the binary collaboration may use this information to execute corresponding transitions or stop the collaboration altogether. Note that this attribute is optional and some document envelope may neither be positive or negative (consider for instance the case of a partial acceptance on a purchase order, where only a few line items are refused, or a back order response). In this case, the business transaction activity is considered successful, again after it has reached a protocol success state.

The isGuaranteedMessageDeliveryRequired refers to the underlying messaging service used to implement the business transaction protocol. The business transaction protocol guarantees the processing of the business documents by the receiving application. However, to achieve this result, the business transaction protocol shall be implemented on top a reliable messaging service that provides guaranteed message delivery at the transport level. In the case where the business transaction does not need to guarantee processing by the receiving application this condition can be relaxed and regular messaging services may be used.

It is important to note that we can only guarantee the complete synchronization of state if reliable messaging is used and if the business transaction is defined to use the request and response acceptance acknowledgement signals, which guarantee that the corresponding business documents were processed by the respective applications.

7.4.3.3 Sample syntax

Here is a slightly more complex transaction with two document flows and three business signals.

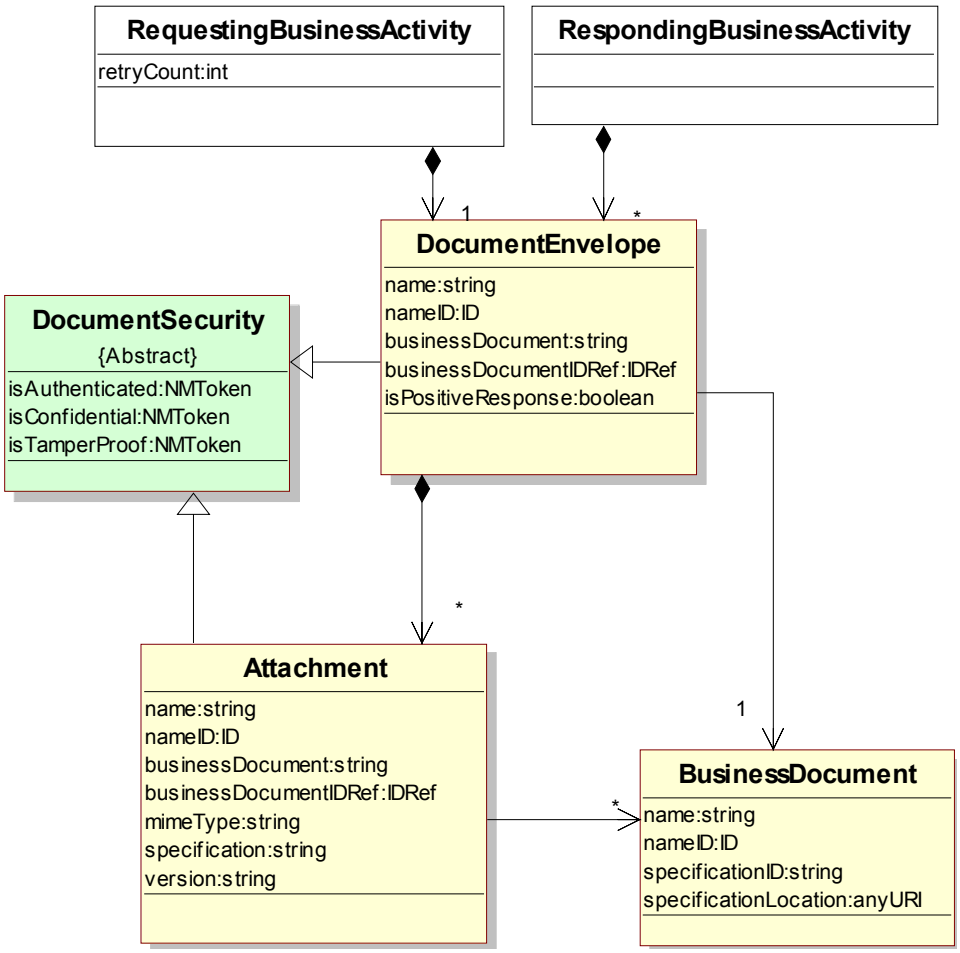
The request requires both receipt and acceptance acknowledgement, the response requires only receipt acknowledgement. “P2D” is a W3C Schema syntax adopted from the ISO 8601 standard and means Period=2 Days. P3D means Period=3 Days, P5D means Period=5 Days. These periods are all measured from original sending of request.

```
<BusinessTransaction
     name="Get Repair Order"
     nameID="122A3DD33"
     isGuaranteedDeliveryRequired="true">
      <RequestingBusinessActivity
           name="Dealer"
           nameID="122A3E833"
           isNonRepudiationReceiptRequired="false"
           isNonRepudiationRequired="false"
           timeToAcknowledgeAcceptance="P3D"
           timeToAcknowledgeReceipt="P2D">
            <DocumentEnvelope
                businessDocument="Get Repair Order"
                businessDocumentIDRef="122A3F613"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity
           name="OEM"
           nameID="122A3E863"
           isNonRepudiationReceiptRequired="false"
           isNonRepudiationRequired="false"
           timeToAcknowledgeReceipt="P1D">
            <DocumentEnvelope
                name="Reject Repair Order"
                isPositiveResponse="false"
                businessDocument="confirmBOD"
                businessDocumentIDRef="122A3F8E3"/>
            <DocumentEnvelope
                name="Accept Repair Order"
                isPositiveResponse="true"
                businessDocument="Show Repair Order"
                businessDocumentIDRef="122A3F6C3"/>
      </RespondingBusinessActivity>
</BusinessTransaction>
```

7.4.3.4 Specifying Business Document flows

Request document flows and response document flows contain Business Documents that pertain to the Business Transaction. The model for this is shown in Figure 8. Business Documents have varying structures. Business signals, however always have the same structure, defined once and for all as part of the ebXML *Business Process Specification Schema*.

911



912

913 **Figure 8: UML Diagram of document flow**

914

915 A document flow is not modeled directly. Rather it is modeled indirectly as
916 a Document Envelope sent by one role and received by the other. The
917 Document Envelope is always associated with one Requesting Business
918 Activity and one Responding Business Activity to model the flow.

919 Document Envelopes are named. There is always only one named
920 Document Envelope for a Requesting Activity. There may be zero, one, or
921 many mutually exclusive, named Document Envelopes for a Responding
922 Activity. For example, the Response Document Envelopes for a purchase
923 order transaction might be named PurchaseOrderAcceptance,
924 PurchaseOrderDenial, and PartialPurchaseOrderAcceptance. In the
925 actual execution of the purchase order transaction, however, only one of
926 the defined possible responses will be sent.

927 The Document Envelope represents the flow of documents between the
928 activities. Each Document Envelope carries exactly one primary Business
929 Document.

A Document Envelope can optionally have one or more attachments, all related to the primary Business Document. The document and its attachments in essence form one transaction in the payload in the ebXML Message Service message structure.

7.4.3.5 Sample syntax

This example shows a business transaction with one request and two possible responses, a success and a failure. The request has an attachment. All the Business Documents are fully qualified with the schema name.

```
<BusinessDocument
        name="confirmBOD"
        nameID="122A3F8E3"
        specificationLocation=”http://www.starstandard/xml/confirmBOD.
xsd”/>
<BusinessDocument
        name="Get Repair Order"
        nameID="122A3F613"
        specificationLocation=”http://www.starstandard/xml/ Get Repair
Order.xsd”/>
<BusinessDocument
        name="Process Repair Order"
        nameID="122A3FCB3"
        specificationLocation=”http://www.starstandard/xml/ Process
Repair Order.xsd”/>
<BusinessDocument
        name="Show Repair Order"
        nameID="122A3F6C3"
        specificationLocation=”http://www.starstandard/xml/ Show Repair
Order.xsd”/>
<BusinessDocument
        name="Update Repair Order"
        nameID="122A3FBE3"
        specificationLocation=”http://www.starstandard/xml/ Update
Repair Order.xsd”/>

<BusinessDocument
        name="Repair Order Details"
        nameID="122A3FBE4"
        specificationLocation=”http://www.starstandard/xml/ Repair Order
Details.xsd”/>


<BusinessTransaction name="Get Repair Order"
                        nameID="122A3DD33"
```

```
                    isGuaranteedDeliveryRequired="true">
        <RequestingBusinessActivity
            name="Dealer"
            nameID="122A3E833"
            isAuthorizationRequired="true"
            isIntelligibleCheckRequired="true"
            isNonRepudiationReceiptRequired="true"
            isNonRepudiationRequired="true"
            timeToAcknowledgeAcceptance=" PT30S"
            timeToAcknowledgeReceipt=" PT10S">
            <DocumentEnvelope
                isAuthenticated="persistent"
                isConfidential="persistent"
                isTamperProof="persistent"
                businessDocument="Get Repair Order"
                businessDocumentIDRef="122A3F613C"/>
         </RequestingBusinessActivity>

        <RespondingBusinessActivity
                name="OEM"
                nameID="122A3E863"
                isAuthorizationRequired="true"
                isIntelligibleCheckRequired="true"
                isNonRepudiationReceiptRequired="true"
                isNonRepudiationRequired="true"
                timeToAcknowledgeReceipt="PT10S">
                <DocumentEnvelope
                    isPositiveResponse="false"
                    isAuthenticated="persistent"
                    isConfidential="persistent"
                    isTamperProof="persistent"
                    businessDocument="confirmBOD"
                    businessDocumentIDRef="122A3F8E3"/>
                <DocumentEnvelope
                    isPositiveResponse="true"
                    isAuthenticated="persistent"
                    isConfidential="persistent"
                    isTamperProof="persistent"
                    businessDocument="Show Repair Order"
                    businessDocumentIDRef="122A3F6C3">
                    <Attachment
                        name=”DeliveryNotes”
                        mimeType=”XML”
                        businessDocument=”Repair Order Details”
                        businessDocumentIDRef="122A3F8E4"
```

```
                                        isConfidential=”none”
                                        isTamperProof=”none”
                                        isAuthenticated=”none”>
                                </Attachment>
                            </DocumentEnvelope>
                        </RespondingBusinessActivity>
                    </BusinessTransaction>
```

7.4.4 Specify a Binary Collaboration

Figure 9 illustrates a binary collaboration.

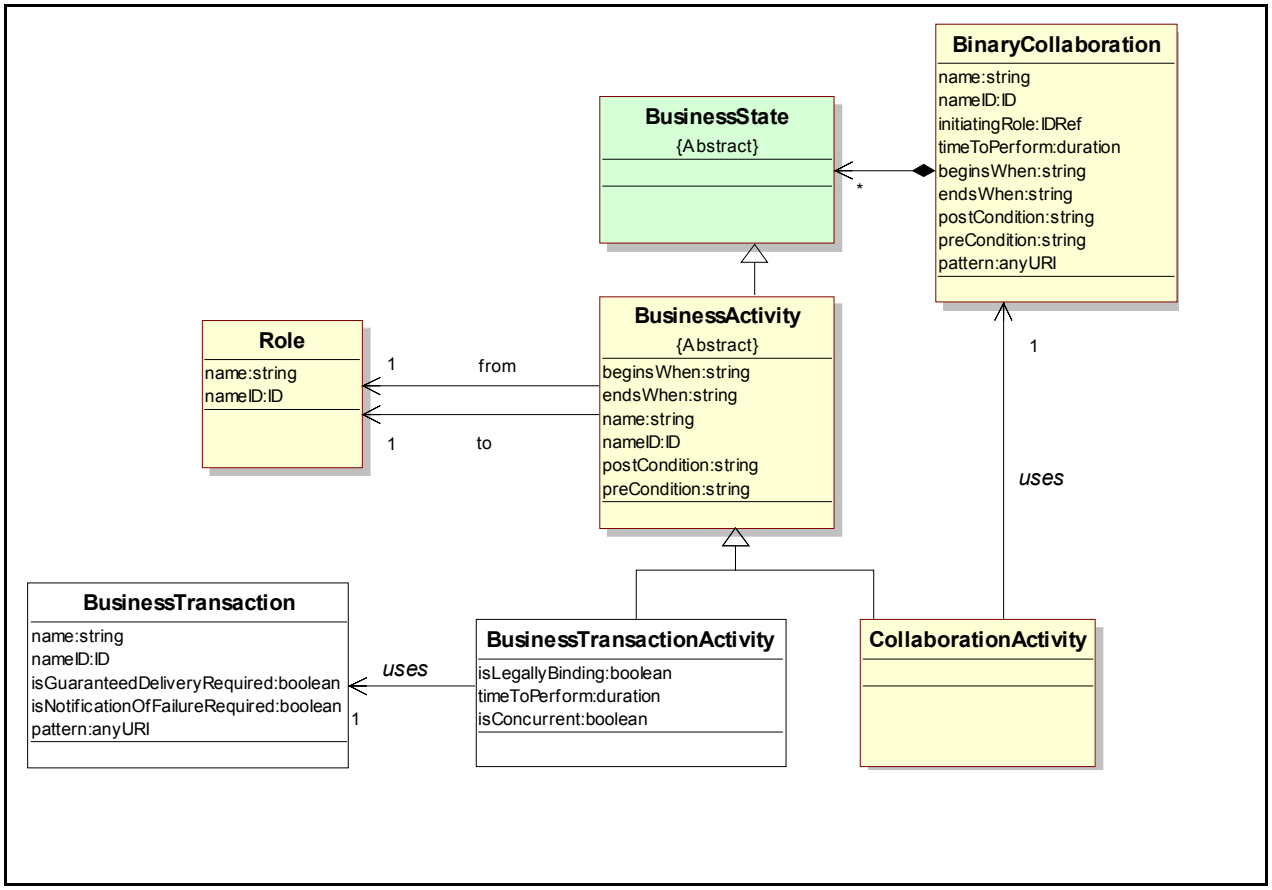


Figure 9: UML Diagram of a Binary Collaboration

7.4.4.1 Key Semantics of a Binary Collaboration

A Binary Collaboration is always between two roles. One of the roles is initiating the collaboration. This is the role, which sends the first message. Note that in certain cases, the initiating role may not be known until runtime when the collaboration choreography starts with a fork, which involves business transaction activities, which can be initiated by the two partners.

A Binary Collaboration consists of one or more Business Activities. These Business Activities are always conducted ***between*** the two Roles of the Binary Collaboration. For each activity one of two roles is assigned to be the InitiatingRole (from) and the other to be the RespondingRole (to). This is irrespective of the Binary Collaboration roles.

A Business Activity can be either a Business Transaction Activity or a Collaboration Activity.

A Business Transaction Activity is the performance of a Business Transaction. Business Transactions are re-useable relative to Business Transaction Activity. The same Business Transaction can be performed by multiple Business Transaction Activities in different Binary Collaborations, or even by multiple Business Transaction Activities in the same Binary Collaboration, sometimes with opposite roles. For instance a “Cancel Purchase Order” Business Transaction could be modeled with two Business Transaction Activities, which can be performed by either party.

A Collaboration Activity is the performance of a Binary Collaboration, within another Binary Collaboration. Binary Collaborations are re-useable relative to Collaboration Activity. The same Binary Collaboration can be performed by multiple Collaboration Activities in different Binary Collaborations, or even by multiple Collaboration Activities in the same Binary Collaboration.

Business Transaction Activity and Collaboration Activity may define business rules with the beginsWhen, endsWhen, preCondition and postCondition attributes. These attributes do not have a specific syntax as part of this specification, so the current type is string. Because these expressions cannot be generally executed by any ebXML infrastructure they are just here as documentation and do not interfere with the choreography of the collaboration. In future specifications they will play a role along with transitions and pseudo-states. The semantics of beginsWhen and endsWhen indicate that the corresponding business activity needs to be started or ended as soon as the expression in the attribute value is true. PreConditions and postConditions indicate that the corresponding business activity may start only if the corresponding expressions are true.

When performing a Binary Collaboration within a Binary Collaboration there is an implicit relationship between the roles at the two levels. Assume that Binary Collaboration X is performing Binary Collaboration Y through Collaboration Activity Q. Binary Collaboration X has the following

roles: Customer and Vendor. In Collaboration Activity Q we assign Customer to be the initiator, and Vendor to be the responder. Binary Collaboration Y has the following roles: Buyer and Seller and a Business Transaction Activity where Buyer is the initiator and Seller the responder. We have now established a role relationship between the roles Customer and Buyer because they are both initiators in activities in the related performing and performed Binary Collaborations.

Since a Business Transaction is atomic in nature, the performing of a single Business Transaction through a Business Transaction Activity is also atomic in nature. If the desired semantic is not atomic, and then the task should be split over multiple transactions. For instance if it is desired to model several partial acceptances of a request, then the request should be modeled as one transaction within a binary collaboration and the partial acceptance(s) as separate transactions.

The CPA/CPP Specification allows that parties agree upon a Collaboration Protocol Agreement (CPA) in order to transact business. A CPA may associate itself with a specific Binary Collaboration. Thus, all Business Transactions performed between two parties should be referenced through Business Transaction Activities contained within a Binary Collaboration.

7.4.4.2 Sample syntax

Here is a simple Binary Collaboration using one of the Business Transactions defined above:

```
<BinaryCollaboration
      name="Repair Order"
      nameID="122A38D93"
      initiatingRoleID=”122A38DA3”
      timeToPerform="P3D">
      <Role
            name="Dealer"
            nameID="122A38DA3"/>
      <Role
            name="OEM"
            nameID="122A38DA3"/>
      <Start
            toBusinessState=”Get Repair Order”
            toBusinessStateIDRef=”122A39C23”
      <BusinessTransactionActivity
            name="Get Repair Order"
            nameID="122A39C23"
            businessTransaction="Get Repair Order"
            businessTransactionIDRef="122A3DD33"
            fromRole="Dealer"
            fromRoleIDRef="122A38DA3"
```

```
                    toRole="OEM"
                    toRoleIDRef="122A38DA3" isConcurrent="true"
                    isLegallyBinding="false"
                    timeToPerform="P1D"/>
               <Failure
                    fromBusinessState="Get Repair Order"
                    fromBusinessStateIDRef="122A39C23"
                    conditionGuard="AnyProtocolFailure"/>
               <Success
                    fromBusinessState="Get Repair Order"
                    fromBusinessStateIDRef="122A39C23"
                    conditionGuard="Success"/>
          </BinaryCollaboration>
```

7.4.5 Specify a MultiParty Collaboration

Figure 10 illustrates a multiparty collaboration

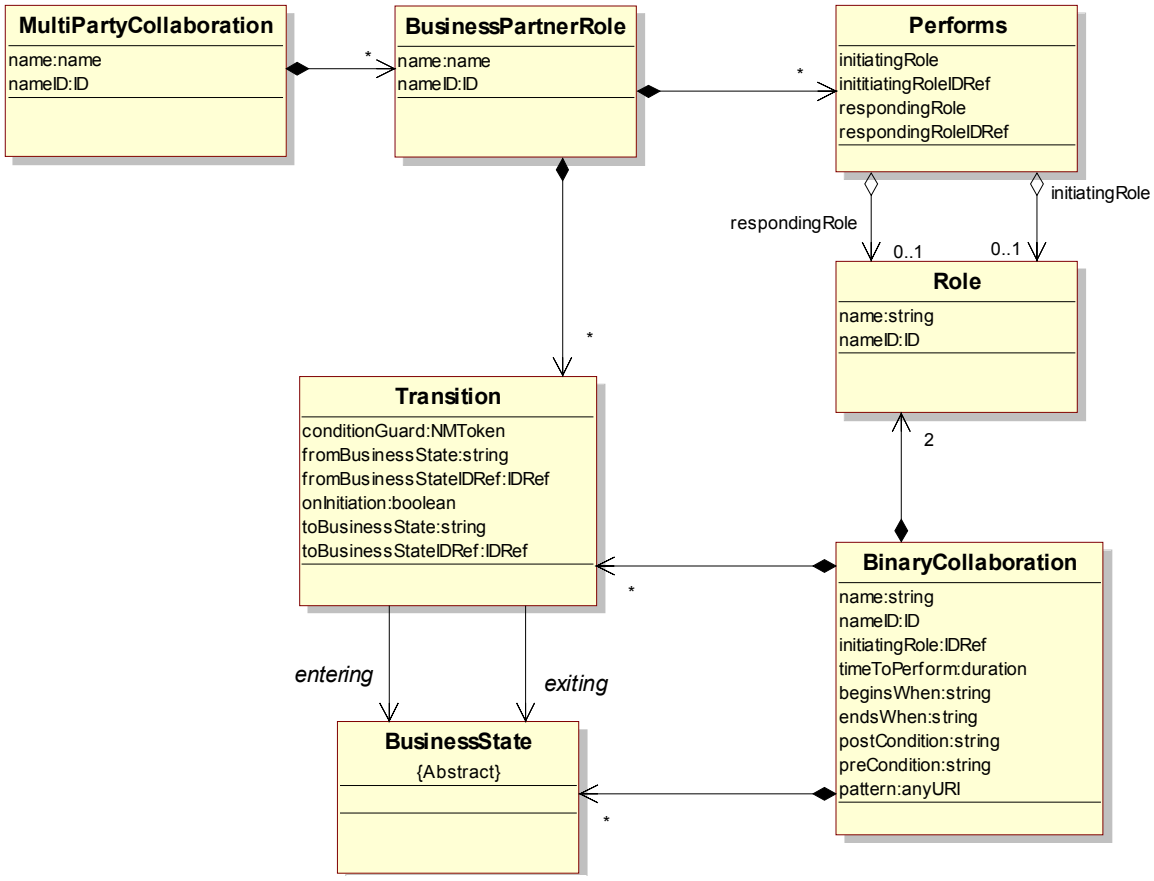


Figure 10: UML Diagram of a MultiParty Collaboration

7.4.5.1 Key Semantics of a Multiparty Collaboration

A Multiparty Collaboration is a synthesis of Binary Collaborations.

A Multiparty Collaboration consists of a number of Business Partner Roles.

Each Business Partner Role performs one Role in one of the binary collaborations, or perhaps one Role in each of several binary collaborations. This is modeled by use of the Performs element.

This 'Performs' linkage between a Business Partner Role and a Role is the synthesis of Binary Collaborations into Multiparty Collaborations. Implicitly the Multiparty Collaboration consists of all the Binary Collaborations in which its Business Partner Roles play Authorized Roles.

Each binary pair of trading partners may be subject to one or more distinct CPAs.

Within a Multiparty Collaboration, you may choreograph transitions between Business Transaction Activities in different Binary Collaborations, as described below.

7.4.5.2 Sample syntax

Here is a simple Multiparty Collaboration which involves 3 parties (Requester, Intermediary and Provider) performing the simple roles of "sender" and "receiver". B is considered an intermediary. The same binary collaborations is executed between each parties: one between the Requester and the Intermediary and the other between the intermediary and the Provider. In this case, the Intermediary plays both roles of the Binary Collaboration.

```
<MultiPartyCollaboration name="Send via intermediary">
      <BusinessPartnerRole name="Requester">
            <Performs role="sender"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="Intermediary">
            <Performs role="receiver"/>
            <Performs role="sender"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="Provider">
            <Performs role="receiver"/>
      </BusinessPartnerRole>
</MultiPartyCollaboration>
```

Note that the role value links the corresponding Binary Collaboration definition to this Multiparty Collaboration definition.

7.4.6 Specify a Choreography

Figure 11 illustrates a choreography.

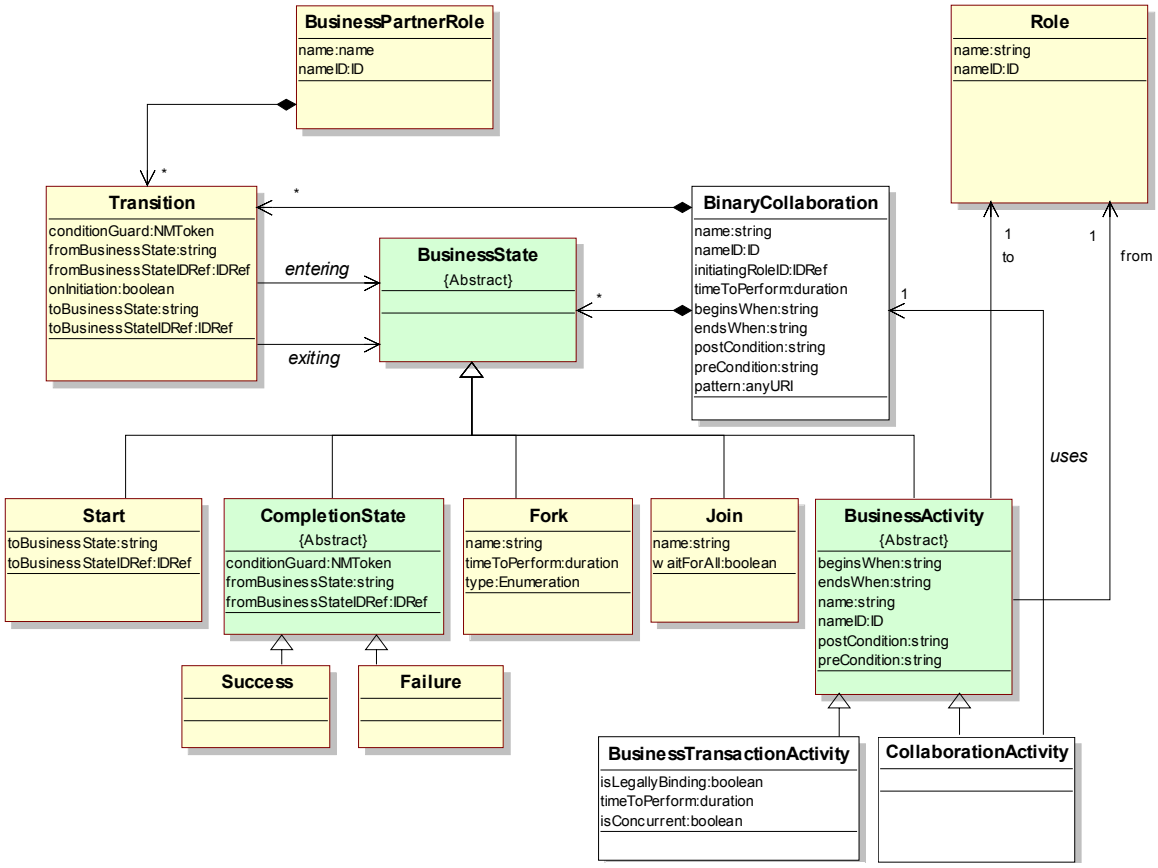


Figure 11: UML Diagram of a Choreography

7.4.6.1 Key Semantics of a Choreography

A Choreography is an ordering and sequencing of Business Activities within a Binary Collaboration.

The choreography is specified in terms of Business States, and transitions between those Business States.

A Business Activity is an abstract kind of Business State. Its two subtypes Business Transaction Activity and Collaboration Activity are concrete Business States. The purpose of a Choreography is to order and sequence Business Transaction Activity and/or Collaboration Activity within a Binary Collaboration, or across Binary Collaborations within a Multiparty Collaboration. The choreography specification and the business transaction protocol specifies unambiguously and at any point in

time which message (DocumentEnvelop or Signal) are expected by any of the parties. A message will either initiate a collaboration or advance its state.

There are a number of auxiliary kinds of Business States that facilitate the choreographing of Business Activities. These include a Start state, a Completion state (which comes in a Success and Failure flavor), a Fork state and a Join state. These are all equivalent to diagramming artifacts on a UML activity chart. A XOR value in the type attribute of a fork means that only one transition coming out of the fork will execute. All the other will become invalid. An ALL value will mean that any business activity pointed to by a transition coming from the fork might be initiated. This business activity may occur in parallel. Similarly, the waitForAll attribute of the join will indicate that all transitions coming into the join shall be executed in order to for the collaboration to reach the join pseudo-state. The timeToPerform attribute on the Fork element may be used to specify that the business activities between the Fork and the Join shall be executed within the specified duration otherwise, the state of the collaboration will automatically advance to the join. This feature is useful in cases where the business activities are optional. For instance a Cancel Purchase Order and Change Purchase Order business transaction activity could be defined as part of a Fork/Join control block. However, most often none of these activity would happen. The semantics of fork and join are such that for instance a fork may be defined without a corresponding join. In this case, the timeToPerform attribute shall not be used. It must only be used in the case where all outgoing transitions from the fork have incoming transitions to the join.

Transitions can originate from Business Transaction Activity or Collaboration Activity within a Binary Collaboration, or from Binary Collaboration within a Multiparty Collaboration. Guards can gate transitions. Guards refer to the status of the Business Transaction Activity from which the transition originates. The guard values include: ProtocolSuccess, AnyProtocolFailure, RequestReceiptFailure, RequestAcceptanceFailure, ResponseReceiptFailure, ResponseAcceptanceFailure, SignalTimeOut, ResponseTimeOut, Failure, BusinessSuccess, BusinessFailure and Success. Transitions may also have a condition expression element. A ConditionExpression element has a language attribute, which specifies in which language the predicate is written. We do not limit the type and number of languages a BSI may support. However, for compliance, a BSI is required to support at least the XPath language, as well as the DocumentEnvelopeLanguage. An XPath expression may involve the content of any DocumentEnvelope received prior to the transition within the scope of the current binary collaboration instance. The DocumentEnvelopeLanguage is simply defined as the name or ID of a document envelope.

The Success and Failure elements represent an aggregation of a state and a transition to this particular state. This transition like regular transitions can be guarded by a conditionGuard. The conditionGuard can be used to indicate that a binary collaboration ends in success or failure

based on the fact that the last business transaction activity response is a business document of a particular type, or based on the content of the response. It is important to note that the success or failure of the collaboration does not affect the success or failure of the individual business transaction activities, which compose the binary collaboration. In particular, the nature of the commitments is not changed when the collaboration ends in a specific state. The success or failure of a collaboration is rather an indication, which can be reported on, or acted upon to initiate other collaborations.

A Transition can also be used to create nested BusinessTransaction-Activities. A nested BusinessTransactionActivity is one where a first transition happens after the receipt of the request in the first transaction, and then the entire second transaction is performed before returning to the first transaction to send the response back to the original requestor. The flag 'onInitiation' in Transition is used for this purpose. Nested BusinessTransactionActivity are typically within a multiparty collaboration. In essence a Role in one Binary Collaboration receives a request, then turns around and becomes the requestor in other Binary Collaboration before coming back and sending the response in the first Binary Collaboration.

isConcurrent is a parameter that governs the flow of transactions. Unlike the security and timing parameters it does not govern the internal flow of a transaction, rather it determines whether multiple instances of that transaction type can be 'open' at the same time as part of the same business transaction activity. IsConcurrent is the parameter that governs this. It is at the business transaction activity level.

7.4.6.2 Sample syntax

Here is the same Binary Collaboration as used before, with choreography added at the end. There is a transition between the two, a start and two possible outcomes of this collaboration, success and failure:

```
<BinaryCollaboration
      name="Repair Order"
      nameID="122A38D93"
      Role=”233A38DA3”
      timeToPerform="P3D">
      <Role
            name="Dealer"
            nameID="122A38DA3"/>
      <Role
            name="OEM"
            nameID="122A38DA3"/>
      <Start
            toBusinessState=” Process Repair Order”
            toBusinessStateIDRef=”122A39C23”/>
```

```
<BusinessTransactionActivity
     name="Process Repair Order"
     nameID="122A39C23"
     businessTransaction="Process Repair Order"
     businessTransactionIDRef="122A39C23"
     fromRole="Dealer"
     fromRoleIDRef="122A38DA3"
     toRole="OEM"
     toRoleIDRef="122A38DA3" isConcurrent="true"
     isLegallyBinding="false"
     timeToPerform="P1D"/>
<BusinessTransactionActivity
     name="Update Repair Order"
     nameID="122A39CA3C3B0172"
     businessTransaction="Update Repair Order"
     businessTransactionIDRef="122A39CA3"
     fromRole="Dealer"
     fromRoleIDRef="122A38DA3"
     toRole="OEM"
     toRoleIDRef="122A38DA3" isConcurrent="true"
     isLegallyBinding="false"
     timeToPerform="P1D"/>
<Success
     fromBusinessState="Process Repair Order"
     fromBusinessStateIDRef="122A39C23"
     conditionGuard="Success"/>
<Success
     fromBusinessState="Update Repair Order"
     fromBusinessStateIDRef="122A39CA3"
     conditionGuard="Success"/>
<Failure
     fromBusinessState="Process Repair Order"
     fromBusinessStateIDRef="122A39C23"
     conditionGuard="AnyProtocolFailure"/>
<Failure
     fromBusinessState="Update Repair Order"
     fromBusinessStateIDRef="122A39CA3"
     conditionGuard="AnyProtocolFailure"/>
<Transition
     onInitiation="false"
     fromBusinessState="Update Repair Order"
     fromBusinessStateIDRef="122A39CA3"
     toBusinessState="Process Repair Order"
     toBusinessStateIDRef="122A39C23"
     conditionGuard="Success">
     <ConditionExpression
          expressionLanguage=
               "DocumentEnvelopeLanguage"
          conditionExpression=
               "Accept Repair Order"/>
```

```
            </Transition>
        </BinaryCollaboration>
```

Optionaly the transition with the condition expression could be expressed with an XPath predicate:

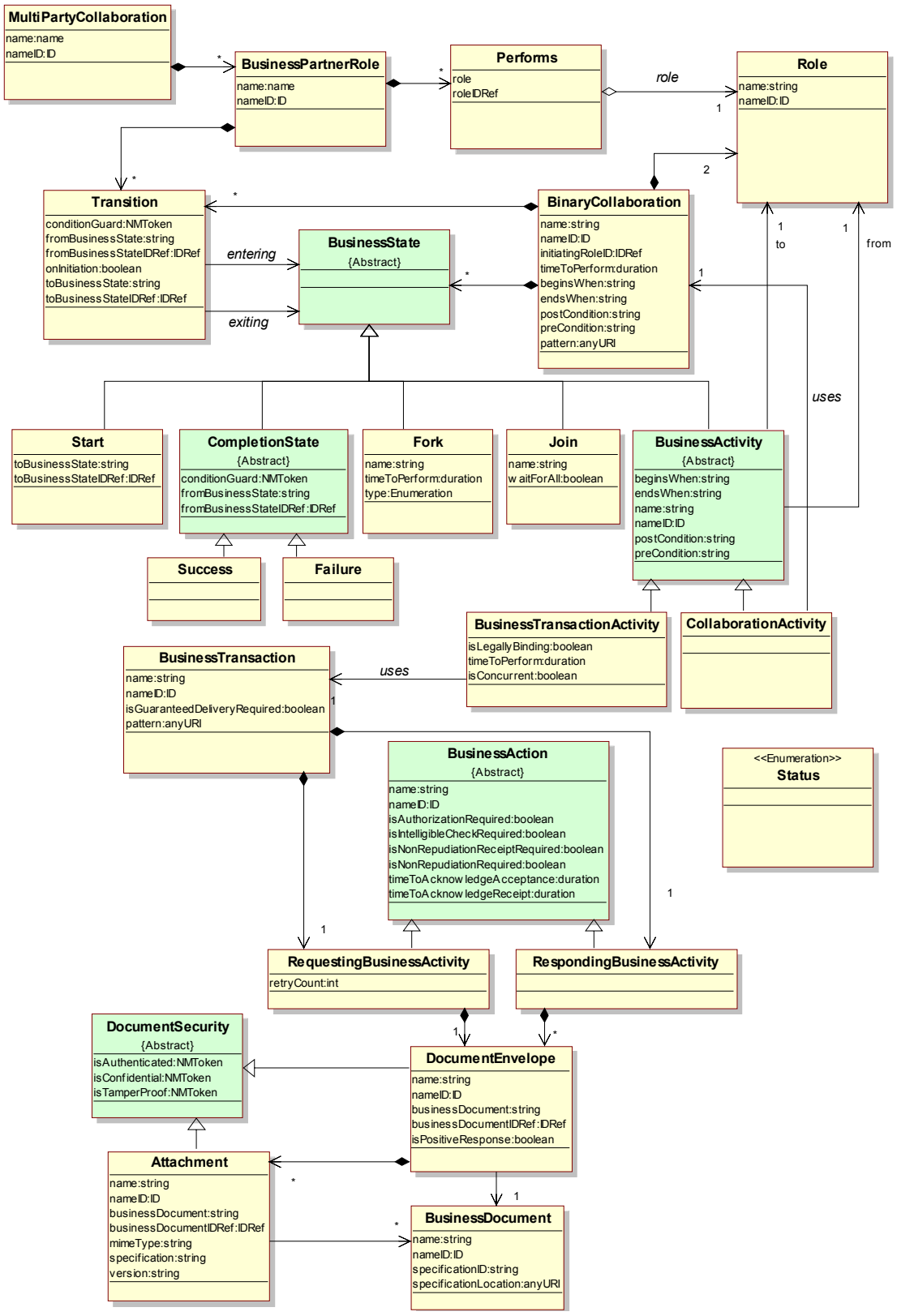
```
            <Transition
                onInitiation="false"
                fromBusinessState="Update Repair Order"
                fromBusinessStateIDRef="122A39CA3"
                toBusinessState="Process Repair Order"
                toBusinessStateIDRef="122A39C23"
                conditionGuard="Success">
                <ConditionExpression
                    expressionLanguage=”XPath”
                    conditionExpression=
                        ”//ConfirmBOD[@status=’Reject’]”
                    />
            </Transition>
```

Here is the same Multiparty Collaboration as defined before, but with a simple choreography (transition) across two Binary Collaborations. BTA is assumed to be the name of the Business Transaction Activity of the Binary Collaboration. Since the same Binary Collaboration is executed between all parties, the transition if from BTA (of the first collaboration) to BTA (of the second collaboration).

```
    <MultiPartyCollaboration name="Send via intermediary">
        <BusinessPartnerRole name="Requester">
            <Performs role="sender"/>
        </BusinessPartnerRole>
        <BusinessPartnerRole name="Intermediary">
            <Performs role="receiver"/>
            <Performs role="sender"/>
            <Transition
                fromBusinessState="BTA"
                toBusinessState="BTA"/>
        </BusinessPartnerRole>
        <BusinessPartnerRole name="Provider">
            <Performs role="receiver"/>
        </BusinessPartnerRole>
    </MultiPartyCollaboration>
```

7.4.7 The whole model

Figure 12 shows the above semantics collectively as a UML class diagram. This diagram contains the whole UML version of the ebXML *Business Process Specification Schema.*



1404

Figure 12: Overall UML Specification Model of the ebXML Business Process Specification Schema

7.5 Core Business Transaction Semantics

The ebXML concept of a business transaction and the semantics behind it are central to predictable, enforceable commerce. It is expected that any Business Service Interface (BSI) will be capable of managing a transaction according to these semantics.

The ebXML Business Transaction semantics allows you to specify electronic commerce transactions that provide

- Interaction Predictability, i.e. have clear roles, clear transaction scope, clear time bounds, clear business information semantics, clear determination of success or failure. Each party can compute without ambiguity and the status of a transaction independently.
- Ability to create Legally Binding Contracts, i.e. the ability to specify that Business Transactions may be agreed to bind the parties.
- Nonrepudiation, i.e. may specify the keeping of artifacts to aid in legal enforceability.
- Authorization Security, i.e. may be specified to require authorization of parties performing roles.
- Document Security, i.e. may be specified to be authorized, authenticated, confidential, tamperproof.
- Reliability, i.e. the ability to specify reliable delivery of Business Documents and signals.
- Run time Business Transaction Semantics, i.e. the rules and configuration parameters required for Business Service Interface software to predictably and deterministically execute ebXML Business Transactions.

Each of the above characteristics of ebXML Business Transaction semantics is discussed in detail below.

7.5.1 Interaction Predictability

All Business Transactions follow a very precisely prescribed flow, or a precisely defined subset there-of. The following is an overall illustration of this flow. It can be thought of as the state machine across the two business partners. The N090R10 chapter on the UMM metamodel has a detail state chart for each of the business partners.
The goal of the Business Transaction Protocol is to synchronize the business state between two parties. As few resources can be shared between company boundaries, we must use such protocol to achieve the business state synchronization as recorded by each party enterprise systems.

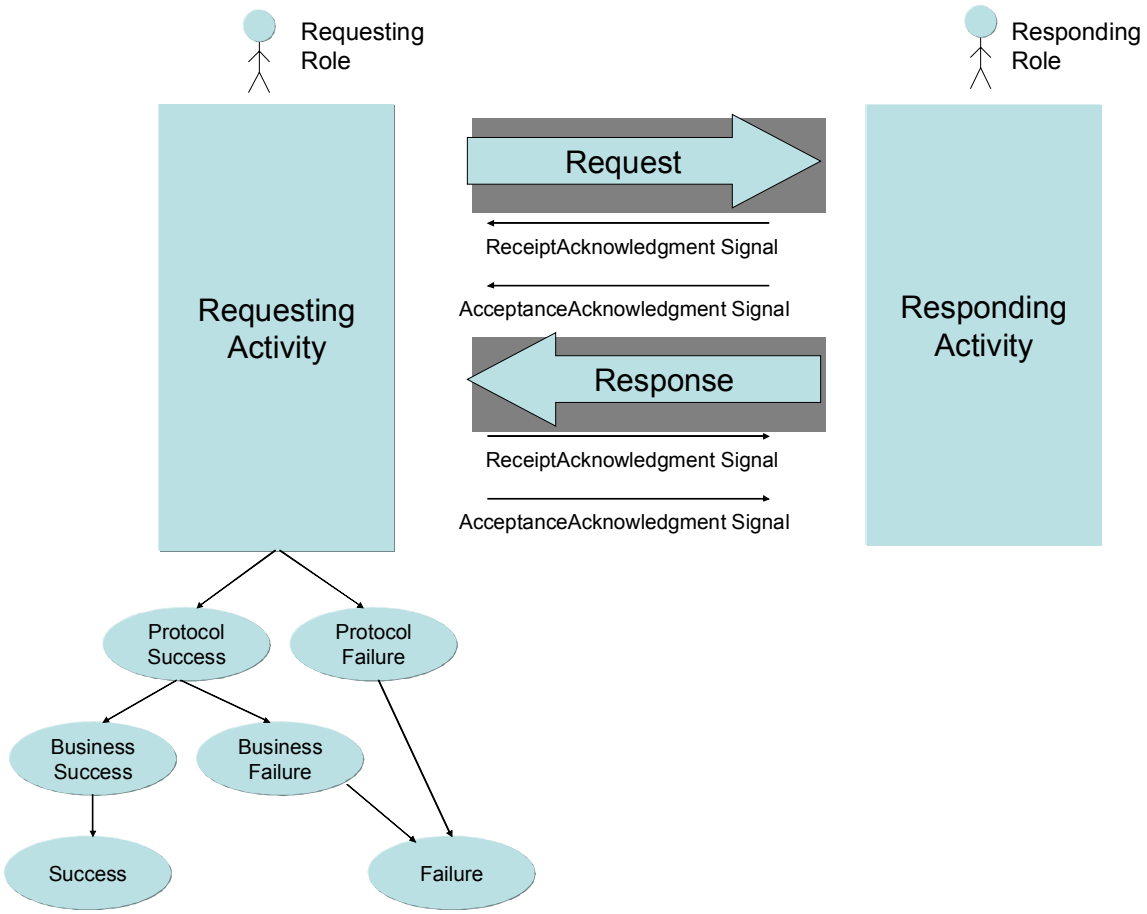


Figure 13: Schematic of core Business Transaction semantics.

In the ebXML model the business transaction always has the following semantics.

1. The Business Transaction is a unit of work. All of the interactions in a business transaction must succeed or each party must not change their state.
2. A Business Transaction is conducted between two business partners playing opposite roles in the transaction. These roles are always the Requesting Role and the Responding Role.
3. A Business Transaction definition specifies exactly when the Requesting Activity is in control, when the Responding Activity is in control, and when control transitions from one to the other. In all Business Transactions control starts at the Requesting Activity, then transitions to the Responding Activity, and then returns to the Requesting Activity.
4. A Business Transaction always starts with a request sent out by the requesting activity.
5. The request serves to transition control to the responding role.

6. After the receipt of the Request document flow, the responding activity may send a receiptAcknowledgement signal and/or an acceptanceAcknowledgement signal to the requesting role.
7. The responding role then enters a responding activity. During or upon completion of the responding activity zero or one response is sent.
8. Control will be returned back to the requesting activity if either a receiptAcknowledgement and/or acceptanceAcknowledgement and/or a response is specified as required. A receiptAcknowledgement (if required) must always occur before an acceptanceAcknowledgement (if required), and an acceptanceAcknowledgement must always occur before a response (if required). Control is returned to the requesting activity based on the last required of these three (if any). If none required, control stays with the responding activity.
9. All business transactions succeed or fail. Success or failure depends on:
    a. The successful transmission of the request, the response and/or receipt and acceptance signals
    b. The occurrence of time-outs
    c. The occurrence of exceptions, as indicated by a negative receipt or acceptance signals
    d. The computation of business failure or success by detecting if the response document was specified – at design time – with isPositiveResponse=false.
10. Both parties can compute the success or failure of the transaction if reliable messaging as well as request and response acceptance acknowledgement signals is used. Once success or failure is thus established, the Business Transaction is considered closed with respect to both parties.
11. Upon receipt of a response the requesting activity may send a receiptAcknowledgement and/or acceptanceAcknowledgement signal back to the responding role. This operation does not pass control back to the responding activity. If the requesting party send the signals after the timeout has occurred, the transaction is considered null and void.
12. Upon identifying a time-out or exception in the processing of a Business Transaction each party will close the transaction and end in a protocol failure state.

7.5.1.1 Transaction Interaction Patterns

The business transaction specification will specify whether a requesting document requires a responding substantive document in order to achieve a "success" end state. In addition, the transaction may specify a proper nonzero time duration for timeToPerform, imposing a deadline for the substantive response.

Furthermore, the specification of a business transaction may indicate, for the request whether receiptAcknowledgement and/or acceptanceAcknowledgement are required, and for the response whether receiptAcknowledgement and/or acceptanceAcknowledgement are required.

The way to specify that a receiptAcknowledgement is required is to set the parameter timeToAcknowledgeReceipt to any proper time duration other than zero. If this parameter has been set to a proper nonzero time duration, optionally either or both of the isIntelligibleCheckRequired and isNonrepudiationOfReceiptRequired parameters may also be set to 'Yes'.

The way to specify that a acceptanceAcknowledgement is required is to set the parameter timeToAcknowledgeAcceptance to any proper time duration other than zero.

So these two acknowledgement related parameters double as Boolean flags for whether the signal is required as part of the transaction, and as values for time-out of the transaction if the signal is not received.

The specification of a business transaction may require each one of these signals independently of whether the other is required. If one is not required, it is actually not allowed. Therefore there is a finite set of combinations. The UMM supplies an illustrative set of patterns representing those combinations, for potential re-use.

7.5.2 Creating legally binding contracts

Trading partners may wish to indicate that a Business Transaction performed as part of an ebXML arrangement is, or is not, intended to be binding. A declaration of intent to be bound is a key element in establishing the legal equivalence of an electronic message to an enforceable signed physical writing. Parties may create explicit evidence of that intent by (1) adopting the ebXML Business Process Specification Schema standard and (2) manipulating the parameter ("isLegallyBinding") designated by the standard to indicate that intent.

In some early electronic applications, trading partners have simply used the presence, or absence, of an electronic signature (such as under the XML-DSIG standard) to indicate that intent. However, documents which rely solely on the presence of a signature may or may not be correctly interpreted, if there is semantic content indicating that a so-called contract is a draft, or nonbinding, or the like.

In ebXML, the presence or absence of an electronic signature cannot indicate by itself determine legally binding assent, because XML-DSIG signatures are reserved for other uses as an assurance of sender identity and message integrity.

isLegallyBinding is a parameter at the BusinessTransactionActivity level, which means that the performing of a BusinessTransaction within a Binary Collaboration is either specified as legally binding or not.

When operating under this standard, parties form binding agreements by exchanging binding messages that agree to terms (e.g., offer and acceptance). The "isLegallyBinding" parameter is Boolean, and its default value is "true." Under this standard, the exclusive manner for indicating that a Business Activity is not intended to be binding is to include a "false" value for the "isLegallyBinding" parameter for the transaction activity. As in EDI, the ebXML standard assumes that Business Transactions are intended by the trading parties to be binding unless otherwise indicated.

As a non-normative matter, parties may wish to conduct nonbinding transactions for a variety of reasons, including testing, and the exchange of proposed offers and counteroffers on a non-committal basis so as to discover a possible agreed set of terms. When using tangible signed documents, parties often do so by withholding a manual signature, or using a "DRAFT" stamp. In ebXML, trading partners may indicate that result by use of the "isLegallyBinding" parameter. See the illustrative Simple Negotiation Pattern set forth in the ebXML E-Commerce Patterns.

7.5.3 Non-Repudiation

Trading partners may wish to conduct legally enforceable business transactions over ebXML. A party may elect to use non-repudiation protocols in order to generate documentation that would assist in the enforcement of the contractual obligation in court, in the case that the counterparty later attempts to repudiate its ebXML Business Documents and messages.

Repudiation generally refers to the ability of a trading partner to argue at a later time, based on the persistent artifacts of a transaction, that it did not agree to the transaction. That argument might be based on assertions that a replying document was not sent, or was not sent by the proper party, or was incorrectly interpreted (under the applicable standard or the trading partners' business rules) as forming agreement.

There are two kinds of non-repudiation protocol available under this document. Each protocol provides the user with some degree of additional evidentiary assurance by creating or requesting additional artifacts that would assist in a later dispute over repudiation issues. Neither is a dispositive absolute assurance. As in the paper world, trading partners are always free to invent colorful new arguments that an apparently-enforceable statement should be ignored. These parameters simply offer some opportunities to make that more difficult.

One imposes a duty on each party to save copies of all Business Documents and Document Envelopes comprising the transaction in the form they where received(e.g. save in encrypted form if they where received in encrypted form) , each on their own side, i.e., requestor saves his request, responder saves his response. This is the isNonRepudiationRequired parameter in the requesting or responding activity. It is logically equivalent to a request that the other trading partner maintain an audit trail. However, failure to comply with that request is not necessarily computationally detectable at run time, nor would it override the determination of a "success" or "failure" end state.

The other requires the receiver of a business document to send a signed receipt, which the original sender saves. This is the isNonRepudiationOfReceiptRequired parameter in the requesting and responding business activity.

NonRepudiationOfReceipt is tied to the ReceiptAcknowledgement, in that it requires the latter to be digitally signed. So NonRepudiationOfReceipt is meaningless if ReceiptAcknowledgement is not required. Failure to comply with NonRepudiation of Receipt would be computationally detectable at run time, and would override the determination of a "failure" end state. If a timeToAcknowledgeReceipt is imposed on a requesting message, and NonRepudiationOfReceipt is true, only a digitally signed receipt will satisfy the imposed timeout deadline. Thus, a failure to send a *signed* receipt within timeToAcknowledgeReceipt, would make the transaction null and void.

7.5.4 Authorization security

Each request or response may be sent by a variety of individuals, representatives or automated systems associated with a business partner. There may be cases where trading partners have more than one ebXML-capable business service interface, representing different levels of authority. In such a case, the parties may establish rules regarding which interfaces or authors may be confidently relied upon as speaking for the enterprise.

In order to invoke those rules, a party may specify IsAuthorizationRequired on a requesting or and responding activity accordingly, with the result that [the activity] will only be processed as valid if the party interpreting it successfully matches the stated identity of the activity's [Authorized Role] to a list of allowed values previously supplied by that party.

IsAuthorizationRequired is specified on the requesting and responding activity accordingly.

7.5.5 Document security

The value of *isConfidential, isTamperProof, isAuthenticated* at the Document Envelope always applies to the primary Business Document. It also applies to each of the attachments unless specifically overridden at the Attachment level. These parameters can have four possible values: none, transient, persistent, transient-and-persistent.

Transient authentication is provided by the communications channel used to transport the *Message*. The specific method will be determined by the communications protocol used.

Persistent authentication means the Business Document signer’s identity shall be verified at the receiving application level.

Transient confidentiality is provided by a secure network protocol, such as SSL as the document is transferred between two adjacent MSH nodes.

Persistent confidentiality is intended to preserve the confidentiality of the message such that only the intended party (application) can see it. The message

shall remain in encrypted form after it is delivered to the MSH node and will be decrypted only by the authorized application. S/MIME can be used to provide that functionality, independent of the transient confidentiality.

Transient isTamperPoof is the ability to detect if the information has been tampered with during transfer between two adjacent MSH nodes.

Persistent isTamperDetectableis the ability to detect if the information has been tampered with after it has been received by MSH, between the MSH and the application.

7.5.6 Reliability

This parameter IsGuaranteedDeliveryRequired at the Business Transaction level states whether guaranteed delivery of the transaction's Business Documents is required.

This is a declaration that trading partners must employ only a delivery channel that provides a delivery guarantee, to send Business Documents in the relevant transaction.

7.5.7 Parameters required for CPP/CPA

The ebXML *Business Process Specification Schema* provides parameters that can be used to specify certain levels of security and reliability. The ebXML *Business Process Specification Schema* provides these parameters in general business terms.

These parameters are generic requirements for the business process, but for ebXML implementations, these parameters are specifically used to instruct the CPP and CPA to require BSI and/or delivery channel capabilities to achieve the specified service levels.

The CPP and CPA translate these into parameters of two kinds.

One kind of parameter determines the selection of certain security and reliability parameters applicable to the transport method and techniques used by the delivery channel. Document security, and Reliability above, are determinators of delivery channel selection.

The other kind of parameter determines the selection of certain service levels or capabilities of the BSI itself, in order for it to support the run time Business Transaction semantics as listed below.

7.6 Run time Business Transaction semantics

The ebXML concept of a business transaction and the semantics behind it are central to predictable, enforceable commerce. It is expected that any Business Service Interface (BSI) will be capable of managing a transaction according to these semantics.

Therefore, the Business Service Interface (BSI), or any software that implements one role in an ebXML collaboration needs at minimum to be able to support the following transaction semantics:

1. Detection of the opening of a transaction
2. Detection of transfer of control
3. Detection of successful completion of a transaction
    a. Application of business rules expressed as isPositiveResponse and transition conditionGuard for determination of success
4. Detection of failed completion of a transaction
    a. Detection of time-outs
    b. Detection of protocol exceptions
    c. Calculation of the received response to identify if it was specified with isPositiveResponse = false

ebXML does not specify how these transaction semantics are implemented but it is assumed that any Business Service Interface (BSI) will be able to support these basic transaction semantics at runtime. If either party cannot provide full support, then the requirements may be relaxed as overrides in the CPP/CPA.

The following sections discuss the two causes of failure: Time-outs and Exceptions. When either one happens, it is the responsibility of the two roles to exit the transaction. It is also expected that the corresponding collaboration will be designed (and choreographed) to execute the appropriate compensating transactions if needed. The responsibilities of the two roles differ slightly and are described in each of the sections below. Generally, if a failure happens at either the responding or requesting role, they will send an exception signal to the other role, and both parties will exit the current transaction.

7.6.1 Timeouts

Since all business transactions must have a distinct time boundary, there are time-out parameters associated with the response, and each of the acknowledgement signals. If the time-out occurs before the corresponding response or signal arrives, the transaction is null and void.

Here are the time-out parameters relative to the three response types:

Response required	Parameter Name	Meaning of timeout

Receipt acknowledgement	timeToAcknowledgeReceipt	The time a responding or requesting role has to acknowledge receipt of a business document.
Acceptance Acknowledgement (Non-substantive)	timeToAcknowledgeAcceptance	The time a responding or requesting role has to non-substantively acknowledge business acceptance of a business document.
Substantive Response	TimeToPerform	The time a responding role has to substantively acknowledge business acceptance of a business document.

A time-out parameter must be specified whenever a requesting partner expects one or more responses to a business document request. A requesting partner must not remain in an infinite wait state.

The time-out value for each of the time-out parameters is absolute i.e. not relative to each other. All timers start when the initial requesting business document is sent. The timer values must comply with the well-formedness rules for timer values.

A BSI needs to comply with the above parameters to detect the appropriate time outs. To preserve the atomic semantics of the Business Transaction, the requesting and responding roles take different action based on time outs.

A responding partner simply terminates if a timeout is thrown. This prevents responding business transactions from hanging indefinitely.

When the time to perform an activity equals the time to acknowledge receipt or the time to acknowledge business acceptance then the highest priority time out exception must be used when the originator provides a reason for revoking their original business document offer. The time to perform exception is lower priority

than both the time to acknowledge receipt and the time to acknowledge business acceptance.

7.6.2 Protocol Exceptions

In addition to timeouts, the Business Transaction protocol provides a series of protocol exception which indicate whether the business processing of the transaction went wrong at either the responding or the requesting role.

7.6.2.1 Receipt Acknowledgement Exception

A *Receipt Exception* signals an error condition in the management of a business transaction. This business signal is returned to the initiating activity that originated the request. This exception must terminate the business transaction. These errors deal with the mechanisms of message exchange such as verification, validation, authentication and authorization and will occur up to message acceptance. Typically the rules and constraints applied to the message will have only dealt with the well-formness of the message.

A receipt exception terminates the business transaction. The following are receipt exceptions:

- Syntax exceptions. There is invalid punctuation, vocabulary or grammar in the business document or business signal.
- Authorization exceptions. Roles are not authorized to participate in the business transaction.
- Signature exceptions. Business documents are not signed for non-repudiation when required.
- Sequence exceptions. The order or type of a business document or business signal is incorrect.

A receipt exception typically means that the current message could not be passed to an application for processing.

7.6.2.2 Acceptance Acknowledgement Exceptions

An Acceptance Exception signals an error condition in a business activity. This business signal is returned to the initiating role that originated the request. This exception must terminate the *business transaction*. These errors deal with the mechanisms that process the *business transaction* and will occur after message verification. Typically the rules and constraints applied to the message will deal with the semantics of message elements and the validity of the request itself. The content is not valid with respect to a responding role's business rules.

An Acceptance Exception terminates the business transaction. The following are business protocol exceptions:

- Business exception. The business rules of the responding activity are violated. The application refused to process the incoming business document.
- Performance exceptions. The requested business action cannot be performed. The application may not be available.

Typically, an Acceptance Exception means that the processing application (usually unknown to the other party) received the corresponding business document but was enable to process them.

A Business Transaction is defined in very atomic and deterministic terms. It always is initiated by the requesting role, and will always conclude at the requesting role. Upon receipt of the required response and/or signals, or time-out of same, the requesting role can unambiguously determine the success or failure of the Business Transaction. A responding role that encounters an Acceptance Exception signals the exception back to the requesting role and then terminates the business transaction.

Conversely, a requesting role that encounters an Acceptance Exception signals the exception back to the responding role and terminates the transaction

A BSI needs to comply specifically with the following parameters to produce the associated special exceptions. The requesting and responding roles take different action as per below.

IsAuthorizationRequired

If a partner role needs authorization to request a business action or to respond to a business action then the sending partner role must sign the business document exchanged and the receiving partner role must validate this business control and approve the authorizer. A responding partner must signal an authorization exception (receipt exception) if the requesting partner role is not authorized to perform the business activity. A sending partner must send notification of failed authorization if a requesting partner is not authorized to perform the responding business activity.

IsNonRepudiationRequired

If non-repudiation of origin and content is required then the business activity must store the business document in its original form for the duration mutually agreed to in a trading partner agreement. A responding partner must signal a receipt exception if the sending partner role has not properly delivered their business document. Similarly, a requesting partner must send receipt exception if a responding partner has not properly delivered their business document.

isNonRepudiationOfReceiptRequired.

Both partners agree to mutually verify receipt of a requesting business document and that the receipt must be non-repudiable. A requesting partner must initiate a notification of failure business

transaction business (possibly revoking a contractual offer) if a responding partner has not properly delivered signed their receipt. For a further discussion of nonrepudiation of receipt, see also the ebXML E-Commerce and Simple Negotiation Patterns.

Non-repudiation of receipt provides the data for the following audit controls.

Verify responding role identity (authenticate) – Verify the identity of the responding role (individual or organization) that received the requesting business document.

Verify content integrity – Verify the integrity of the original content of the business document request.

isPositiveResponse

An expression whose evaluation results in TRUE or FALSE. If TRUE this DocumentEnvelope is intended as a positive response to the request. If isPositiveResponse = FALSE, the business transaction activity ends in business failure mode. The value for this parameter supplied for a DocumentEnvelope is an assertion by the sender of the DocumentEnvelope regarding its intent for the transaction to which it relates, but does not bind the recipient, or override the computation of transactional success or failure.

7.6.3 Computation of the status of a Business Transaction Activity

The following figure represent the computation of the success or failure of a business transaction activity based on the different possible scenarios.

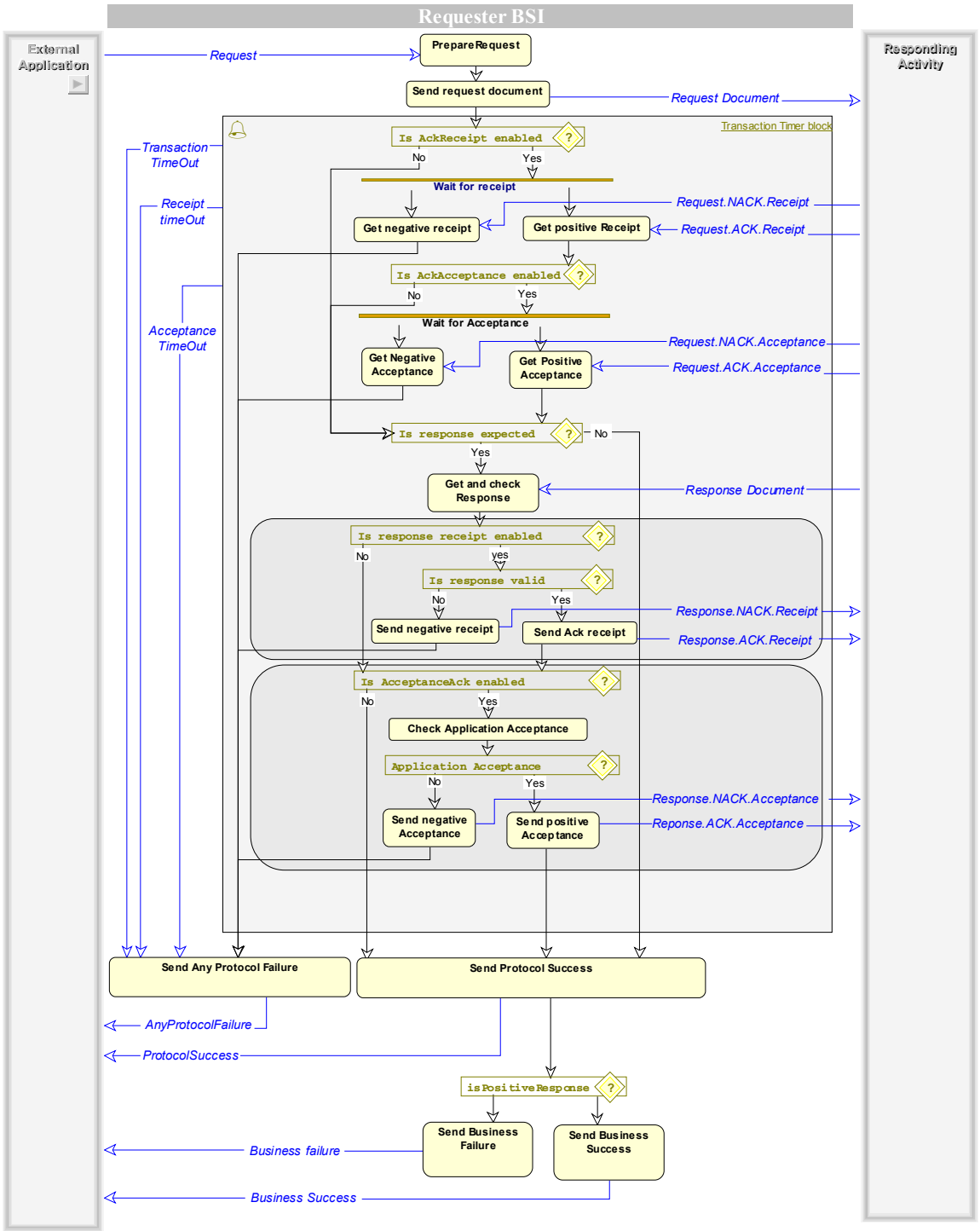


Fig 14. Computation of the Status of a Business Transaction Activity

The values of the enumeration of the state of a business transaction of the conditionGuard on a transition are:

- ProtocolSuccess
- AnyProtocolFailure

1850 - RequestReceiptFailure
1851 - RequestAcceptanceFailure
1852 - ResponseReceiptFailure
1853 - ResponseAcceptanceFailure
1854 - SignalTimeout
1855 - ResponseTimeout
1856 • BusinessSuccess (isPositiveResponse=true or no isPositiveResponse
1857 attribute)
1858 • BusinessFailure(isPositiveResponse=false)
1859 • Success (both protocol and business success)
1860 • Failure (AnyProtocolFailure or BusinessFailure).
1861
1862 BusinessFailure assumes that the transaction was successful from a “protocol”
1863 perspective, meaning that the state between the two parties could be effectively
1864 synchronized. However, the intent of the response was negative with respect to
1865 the request. As we mentioned earlier, this is an optional qualification of the
1866 response, agreed upon at design time, and some messages may not be
1867 qualifiable, i.e. they are neither positive or negative. The way business document
1868 specifications are designed allows to define two “logicial” documents from the
1869 same physical document and a condition expression evaluated at run-time by the
1870 BSI. If the condition is true and isPositiveResponse = false, then the transaction
1871 ends in business failure based on the business document content. Of course
1872 entire documents can be tagges with isPositiveResponse=false, not just when
1873 they contain a particular field value.
1874
1875 It is required that each business transaction activity be designed such that there
1876 is at a minimum two transitions from the business transaction activity, one with a
1877 conditionGuard with a Success value, the other one with a Failure value. Even if
1878 in case of failure the transitions goes to the failure state of the collaboration.

1879 *7.7 Runtime Collaboration Semantics*

1880 The ebXML collaboration semantics contain a number of relationships between
1881 multiparty collaborations and binary collaborations, between recursive layers of
1882 binary collaborations, and choreographies among transactions in binary
1883 collaborations. It is anticipated that over time BSI software will evolve to the point
1884 of monitoring and managing the state of a collaboration, similar to the way a BSI
1885 today is expected to manage the state of a transaction. For the immediate future,
1886 such capabilities are not expected and not required.

1887 *7.8 Where the ebXML Business Process Specification Schema* 1888 *May Be Implemented*

1889 The ebXML *Business Process Specification Schema* should be used wherever
1890 software is being specified to perform a role in an ebXML business collaboration.
1891 Specifically, the ebXML *Business Process Specification Schema* is intended to
1892 provide the business process and document specification for the formation of
1893 ebXML trading partner Collaboration Protocol Profiles and Agreements.

However, the ebXML *Business Process Specification Schema* may be used to specify any electronic commerce collaboration. It may also be used for non-commerce collaborations, for instance in defining transactional collaborations among non-profit organizations or internally in enterprises.

Every BSI which is in the position of sending a signal or a document envelop shall verify if sending this message will violate the business transaction definitions and shall not send it if such a condition is detected. For instance sending a signal or a response after a timeout has occurred is prohibited. Similarly, sending a receipt on a document envelop which do not have the same digest as the original document envelop is prohibited. Rather, the BSI should send an exception back to the application initiated the particular message.

Compliant BSIs do not need to support multiparty collaborations. It will be required for further version of the specification.

As of the current version, it is not requested that BSI be able to support multi-party collaboration. The current specification does not support the notions of context and correlation, which makes in hard to implement but for simple cases. Future releases of this specification will provide the corresponding definitions.

7.9 Guidelines for Business Service Interface Interoperability

We have taken great care in this new version of the specification to distinguish what is executable and computable versus general expressions written in text and associated with model elements. In particular, we exclude, beginsWhen, endsWhen, preCondition and postCondition from the responsibility of a BSI. In the future these model constructs will become the responsibility of the BSI as we identify an executable syntax.

Another important point for interoperability is that the context of a binary collaboration is limited to the document flows that are received or sent by the BSI. In no case, the BSI could be in charge in looking up information in other systems, internal or external to determine the result of condition expressions.

A BSI is required to support two languages for the ConditionExpression element: the XPath language, as well as the “DocumentEnvelopeLanguage”. An XPath expression may involve the content of any DocumentEnvelope received prior to the transition within the scope of the current binary collaboration instance. The “DocumentEnvelopeLanguage” is simply defined as the name or ID of a document envelope.

7.10 Collaboration and transaction well-formedness rules

The following rules should be used in addition to standard parsing to properly constrain the values of the attributes of the elements in an ebXML Business Process Specification.

Business Transaction

[0] If non-repudiation is required then the input or returned business document must be a tamper-proofed entity.

[1] If authorization is required then the input business document and business signal must be an authenticated or a tamper proofed secure entity.

[2] The time to acknowledge receipt must be less than the time to acknowledge acceptance if both properties have values.

timeToAcknowledgeReceipt < timeToAcknowledgeAcceptance

[3] If the time to acknowledge acceptance is null then the time to perform an activity must either be equal to or greater than the time to acknowledge receipt.

[4] The time to perform a transaction cannot be null if either the time to acknowledge receipt or the time to acknowledge acceptance is not null.

[5] If non-repudiation of receipt is required then the time to acknowledge receipt cannot be null.

[6] The time to acknowledge receipt, time to acknowledge acceptance and time to perform cannot all be zero.

RequestingBusinessActivity

[7] There must be one input transition whose source state vertex is an initial pseudo state.

[8] There must be one output transition whose target state vertex is a final state specifying the state of the machine when the activity is successfully performed.

[9] There must be one output transition whose target state vertex is a final state specifying the state of the machine when the activity is NOT successfully performed due to a process control exception.

[10] There must be one output transition whose target state vertex is a final state specifying the state of the machine when the activity is NOT successfully performed due to a business process exception.

[11] There must be one output document flow from a requesting business activity that in turn is the input to a responding business activity.

[12] There must be zero or one output document flow from a responding business activity that in turn is the input to the requesting business activity.

RespondingBusinessActivity

[13] There must be one input transition from a document flow that in turn has one input transition from a requesting business activity.

[14] There must be zero or one output transition to a document flow that in turn has an output transition to a requesting business activity.

Business Collaboration

[15] A Business Partner Role cannot provide both the roles of the same business transaction activity.

8 ebXML Business Process Specification Schema –

In this section we describe the XML Schema version of the Specification Schema.

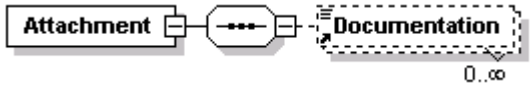
- An example XML Business Process Specification listed in Appendix A
- A table listing all the elements with definitions and parent/child relationships
- A table listing all the elements, each with a cross reference to the corresponding class in the UML version of the specification schema
- Rules about namespaces and element references

8.1 Documentation for the Schema

This section will document the Schema. The Schema has been derived from the UML model. The correlation between the UML classes and Schema elements will be shown separately later in this document.

Elements:	Complex types	Attr. groups
Attachment	**BusinessAction**	**documentSecurity**
AttributeSubstitution	**BusinessActivity**	**name**
BinaryCollaboration	**RoleType**	
BusinessDocument		
BusinessPartnerRole		
BusinessTransaction		
BusinessTransactionActivity		
CollaborationActivity		
ConditionExpression		
Documentation		
DocumentEnvelope		
DocumentSubstitution		
Failure		
Fork		
Include		
Role		
Join		
MultiPartyCollaboration		
Package		
Performs		
ProcessSpecification		
RequestingBusinessActivity		
RespondingBusinessActivity		
Start		
SubstitutionSet		
Success		
Transition		

8.1.1 Element Attachment

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	An optional attachment to a BusinessDocument in a DocumentEnvelope. Recommendation: Either use businessDocument +businessDocumentIDRef attributes OR use specification +mimeType attributes.
Children	**Documentation**
Used by	element **DocumentEnvelope**

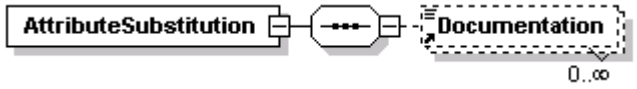
Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the attachment.
nameID	xsd:ID			XML ID version of name
businessDocument	xsd:string	required		A BusinessDocument can define an Attachment's type. If it is not of a defined Business Document, the mime type and specification attribute will be the only indication of its type.
businessDocumentIDRef	xsd:IDREF			The XML IDREF version of businessDocument
specification	xsd:anyURI			A reference to an external source of description of this attachment.
mimeType	xsd:string	optional		Defines the valid MIME (Multipurpose Internet Mail Extensions) type of this Attachment. Example: 'application/pdf'
isAuthenticated	xsd:NMTOKEN		none	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.(See also section on Document Security)
isConfidential	xsd:NMTOKEN		none	The information entity is encrypted so that unauthorized parties cannot view the information(See also section on Document Security)
isTamperDetectable	xsd:NMTOKEN		none	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.(See also section on Document Security)

source

```
<xsd:element name="Attachment">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="businessDocument" type="xsd:string" use="required"/>
  <xsd:attribute name="businessDocumentIDRef" type="xsd:IDREF"/>
  <xsd:attribute name="specification" type="xsd:anyURI"/>
  <xsd:attribute name="mimeType" type="xsd:string" use="optional"/>
  <xsd:attributeGroup ref="documentSecurity"/>
 </xsd:complexType>
</xsd:element>
```

2003 8.1.2 Element: AttributeSubstitution

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	Attribute Substitution specifies that an attribute value should be used in place of some attribute value in an existing process specification.
Children	**Documentation**
Used by	element **SubstitutionSet**

Attributes

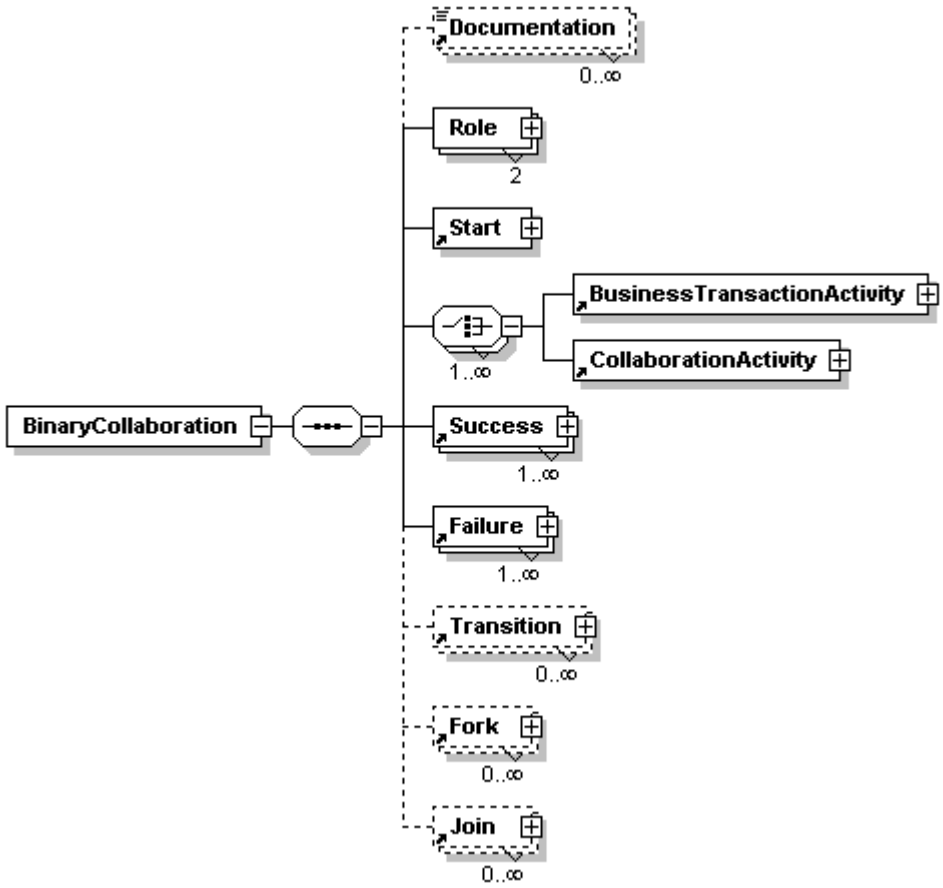
Name	Type	Use	Default	Annotation
attributeName	xsd:string	required		The name of an attribute of any element within the scope of the substitution set.
value	xsd:string	required		The value, which shall replace the current value of the attribute.

Source

```
<xsd:element name="AttributeSubstitution">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="attributeName" type="xsd:string" use="required"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
 </xsd:complexType>
</xsd:element>
```

2004

8.1.3 Element: BinaryCollaboration

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	Binary Collaboration defines a protocol of interaction between two roles. One must be the initiating role, and one the responding role. Binary Collaboration is a choreographed set of states among collaboration roles. The activities of performing business transactions or other collaborations are a kind of state. Binary Collaboration choreographs one or more business transaction activities between two roles. Binary Collaboration is not an atomic transaction. A binary collaboration may be used within another binary collaboration via a collaboration activity.
Children	Documentation Role Start BusinessTransactionActivity CollaborationActivity Transition Success Failure Fork Join
Used by	elements Package ProcessSpecification

Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the Binary Collaboration.
nameID	xsd:ID			The XML ID version of name Recommendation: name ID should be globally unique identifier.
pattern	xsd:anyURI			The optional reference to a pattern that this binary collaboration is based on.
beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.

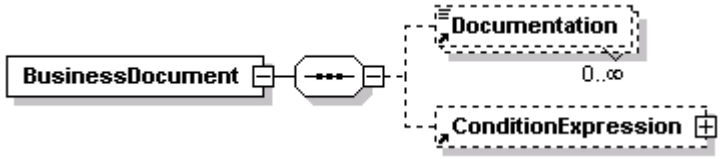
	postCondition	xsd:string		A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration.
	timeToPerform	xsd:duration		The period of time, starting upon initiation of the first activity, within which this entire collaboration must conclude.
	initiatingRoleID	xsd:IDREF	required	Reference to the role that initiates the transaction.

source

```
<xsd:element name="BinaryCollaboration">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element name="Role" type="RoleType" minOccurs="2" maxOccurs="2"/>
   <xsd:element ref="Start"/>
   <xsd:choice maxOccurs="unbounded">
    <xsd:element ref="BusinessTransactionActivity"/>
    <xsd:element ref="CollaborationActivity"/>
   </xsd:choice>
   <xsd:element ref="Success" maxOccurs="unbounded"/>
   <xsd:element ref="Failure" maxOccurs="unbounded"/>
   <xsd:element ref="Transition" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Fork" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Join" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="pattern" type="xsd:anyURI"/>
  <xsd:attribute name="beginsWhen" type="xsd:string"/>
  <xsd:attribute name="endsWhen" type="xsd:string"/>
  <xsd:attribute name="preCondition" type="xsd:string"/>
  <xsd:attribute name="postCondition" type="xsd:string"/>
  <xsd:attribute name="timeToPerform" type="xsd:duration"/>
  <xsd:attribute name="initiatingRoleID" type="xsd:IDREF" use="required"/>
 </xsd:complexType>
</xsd:element>
```

8.1.4 Element: BusinessDocument

Diagram



Namespace	http://www.ebxml.org/BusinessProcess
Description	BusinessDocument is a generic name of a document. A BusinessDocument may have one Condition Expression. This determines whether this is a valid business document for its envelope.
Children	**Documentation** **ConditionExpression**
Used by	elements **Package** **ProcessSpecification**

Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the generic name of the Business
nameID	xsd:ID			XML ID version of name
specificationLocation	xsd:anyURI			Reference to an external source of the schema

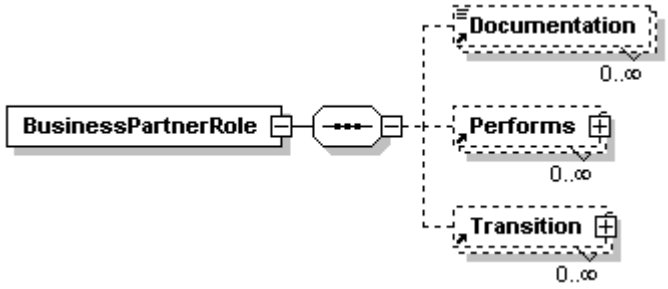
	specificationID xsd:anyURI	Reference to an external source of the schema definition. This defines the absolute path including the element id within the schema definition that defines the type of this document. Use either specificationLocation or specificationID.

source

```
<xsd:element name="BusinessDocument">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ConditionExpression" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="specificationLocation" type="xsd:anyURI"/>
  <xsd:attribute name="specificationID" type="xsd:anyURI"/>
 </xsd:complexType>
</xsd:element>
```

8.1.5 Element: BusinessPartnerRole

Diagram



Namespace: http://www.ebxml.org/BusinessProcess

Description: A BusinessPartnerRole is the role played by a business partner in a MultiPartyCollaboration. A BusinessPartnerRole performs at most one Role in each of the Binary Collaborations that make up the Multiparty Collaboration.

Wellformedness Rule: A partner must not perform both roles in a given business activity

Children: **Documentation** **Performs** **Transition**

Used by: element **MultiPartyCollaboration**

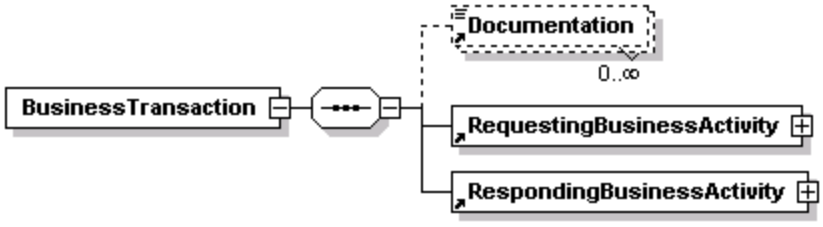
Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the role played by a partner in the overall multiparty business collaboration, e.g. customer or supplier.
nameID	xsd:ID			The XML ID version of name

Source

```
<xsd:element name="BusinessPartnerRole">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Performs" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Transition" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
 </xsd:complexType>
</xsd:element>
```

8.1.6 Element: BusinessTransaction

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	A business transaction is a set of business information and business signal exchanges amongst two commercial partners that must occur in an agreed format, sequence and time period. If any of the agreements are violated then the transaction is terminated and all business information and business signal exchanges must be discarded. Business Transactions can be formal as in the formation of on-line offer/acceptance commercial contracts and informal as in the distribution of product announcements. A BusinessTransaction can be performed by many BusinessTransactionActivites. A BusinessTransaction has exactly one RequestingBusinessActivity. A BusinessTransaction has exactly one RespondingBusinessActivity
Children	**Documentation** **RequestingBusinessActivity** **RespondingBusinessActivity**
Used by	elements **Package** **ProcessSpecification**

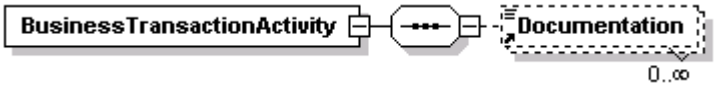
Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the Business Transaction.
nameID	xsd:ID			The XML ID version of name. Recommendation: name ID should be globally unique identifier
pattern	xsd:anyURI			The optional reference to a pattern that this transaction is based on
isGuaranteedDeliveryRequired	xsd:boolean		false	Both partners must agree to use a transport that guarantees delivery

Source

```
<xsd:element name="BusinessTransaction">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="RequestingBusinessActivity"/>
   <xsd:element ref="RespondingBusinessActivity"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="pattern" type="xsd:anyURI"/>
  <xsd:attribute name="isGuaranteedDeliveryRequired" type="xsd:boolean" default="false"/>
 </xsd:complexType>
</xsd:element>
```

8.1.7 Element: BusinessTransactionActivity

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
type	extension of **BusinessActivity**
Description	A business transaction activity defines the use of a business transaction within a binary collaboration. A business transaction activity is a business activity that executes a specified business transaction. More than one instance of the same business transaction activity can be open at one time if the isConcurrent property is true. A Role may not be both the requestor and the responder in a business transaction.

Children	**Documentation**
Used by	element **BinaryCollaboration**

Attributes

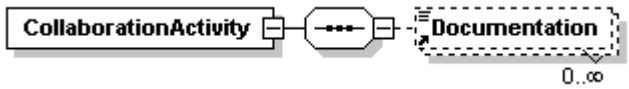
Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the activity uniquely within the binary collaboration
nameID	xsd:ID			The XML ID version of name
businessTransaction	xsd:string	required		A reference, by name to the Business Transaction performed by this Business Transaction Activity
businessTransactionIDRef	xsd:IDREF			The XML IDREF version of businessTransaction
fromRole	xsd:string	required		The name of the initiating role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the requestor in the BusinessTransaction performed by this activity
fromRoleIDRef	xsd:IDREF			The XML IDREF version of fromAuthorizedRole
toRole	xsd:string	required		The name of the responding role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the responder in the BusinessTransaction performed by this activity
toRoleIDRef	xsd:IDREF			The XML IDREF version of toAuthorizedRole
isConcurrent	xsd:boolean		true	If the BusinessTransactionActivity is concurrent then more than one instance of the associated BusinessTransaction can be open the same time as part of the execution of this BusinesTransactionActivity
isLegallyBinding	xsd:boolean		true	Defines whether the Business Transaction performed by this activity is intended by the trading parties to be binding. Default value is True.
timeToPerform	xsd:duration			The period of time, starting upon the sending of the request, within which both partners agree to conclude the business transaction executed by this Business Transaction Activity.
beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.
postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration.

Source

```
<xsd:element name="BusinessTransactionActivity">
 <xsd:complexType>
  <xsd:complexContent>
   <xsd:extension base="BusinessActivity">
    <xsd:sequence>
     <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="businessTransaction" type="xsd:string" use="required"/>
    <xsd:attribute name="businessTransactionIDRef" type="xsd:IDREF"/>
    <xsd:attribute name="isConcurrent" type="xsd:boolean" default="true"/>
    <xsd:attribute name="isLegallyBinding" type="xsd:boolean" default="true"/>
    <xsd:attribute name="timeToPerform" type="xsd:duration"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
```

2017

8.1.8 Element: CollaborationActivity

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
type	extension of **BusinessActivity**
Description	A collaboration activity is the activity of performing a binary collaboration within another binary collaboration
Children	**Documentation**
Used by	element **BinaryCollaboration**

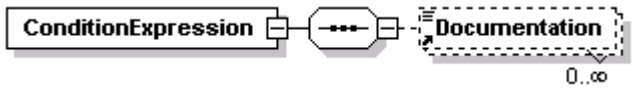
Attributes

Name	Type	Use	Default	Annotation
nameID	xsd:ID			ID of the name
name	xsd:string	required		Defines the name of the activity uniquely within the binary collaboration
fromRole	xsd:string	required		The name of the initiating role in the Collaboration Activity. This must match one of the AuthorizedRoles in the parent binary collaboration and will become the initiator in the BinaryCollaboration performed by this activity
fromRoleIDRef	xsd:IDREF			The XML IDREF version of fromAuthorizedRole
toRole	xsd:string	required		The name of the responding role in the Collaboration Activity. This must match one of the AuthorizedRoles in the parent binary collaboration and will become the responder in the BinaryCollaboration performed by this activity
toRoleIDRef	xsd:IDREF			The XML IDREF version of toAuthorizedRole
binaryCollaboration	xsd:string	required		A reference, by name, to the Binary Collaboration performed by this Collaboration Activity
binaryCollaborationIDRef	xsd:IDREF			The XML IDREF version of binaryCollaboration
beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.
postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration.

Source

```
<xsd:element name="CollaborationActivity">
 <xsd:complexType>
  <xsd:complexContent>
   <xsd:extension base="BusinessActivity">
    <xsd:sequence>
     <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="binaryCollaboration" type="xsd:string" use="required"/>
    <xsd:attribute name="binaryCollaborationIDRef" type="xsd:IDREF"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
```

8.1.9 Element: ConditionExpression

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	Condition Expression is an expression that can be evaluated to TRUE or FALSE.
Children	**Documentation**
Used by	elements **BusinessDocument Failure Success Transition**

Attributes

Name	Type	Use	Default	Annotation
expressionLanguage	xsd:string	required		The language of the expression, e.g. XPATH
expression	xsd:string	required		An expression whose evaluation results in TRUE or FALSE. For a transition, this determines whether this transition should happen or not. For a business document, this determines whether this is a valid business document for its envelope. The expression can refer to the name or content of the most recent DocumentEnvelope or content of documents within it.

Source

```
<xsd:element name="ConditionExpression">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="expressionLanguage" type="xsd:string" use="required"/>
  <xsd:attribute name="expression" type="xsd:string" use="required"/>
 </xsd:complexType>
</xsd:element>
```

8.1.10 Element: Documentation

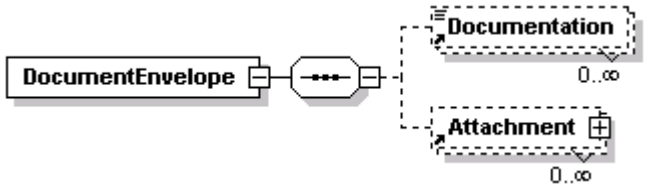
Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
type	extension of **xsd:string**
Description	Defines user documentation for any element. Must be the first element of its container. Documentation can be either inline PCDATA and/or a URI to where more complete documentation is to be found
Used by	elements **Attachment AttributeSubstitution BinaryCollaboration BusinessDocument BusinessPartnerRole BusinessTransaction BusinessTransactionActivity CollaborationActivity ConditionExpression DocumentEnvelope DocumentSubstitution Failure Fork Include Join MultiPartyCollaboration Package Performs ProcessSpecification Start SubstitutionSet Success Transition** complexTypes **BusinessAction RoleType**

Source

```
<xsd:element name="Documentation">
 <xsd:complexType>
  <xsd:simpleContent>
   <xsd:extension base="xsd:string">
    <xsd:attribute name="uri" type="xsd:anyURI"/>
   </xsd:extension>
  </xsd:simpleContent>
 </xsd:complexType>
</xsd:element>
```

2028
2029

2030 8.1.11 Element: DocumentEnvelope

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	A DocumentEnvelope is what conveys business information between the two roles in a business transaction. One DocumentEnvelope conveys the request from the requesting role to the responding role, and another DocumentEnvelope conveys the response (if any) from the responding role back to the requesting role. A documentEnvelop contains exactly one primary Business document. It contains an optional set of attachments related to primary document. Wellformedness Rules: A Document Envelope is associated with exactly one requesting and one responding activity. IsPositiveResponse is not a relevant parameter on a DocumentEnvelope sent by a requesting activity
Children	**Documentation** **Attachment**
Used by	elements **RequestingBusinessActivity** **RespondingBusinessActivity**

attributes

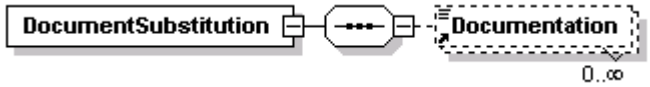
Name	Type	Use	Default	Annotation
name	xsd:string			Defines Name of the DocumentEnvelope
nameID	xsd:ID			Defines ID of the DocumentEnvelope
businessDocument	xsd:string	required		The name of the business document.
businessDocumentIDRef	xsd:IDREF			The XML IDREF version of businessDocument
isPositiveResponse	xsd:boolean			TRUE or FALSE. If TRUE this DocumentEnvelope is intended as a positive response to the request. The value for this parameter supplied for a DocumentEnvelope is an assertion by the sender of the DocumentEnvelope regarding its intent for the transaction to which it relates, but does not bind the recipient, or override the computation of transactional success or failure using the transaction's guard expressions. In some situations this could be an XPath expression that interrogates the BusinessDocument in the envelope. IsPositiveResponse is only relevant for responses, and is ignored in requests.
isAuthenticated	xsd:NMTOKEN		none	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".
isConfidential	xsd:NMTOKEN		none	The information entity is encrypted so that unauthorized parties cannot view the information.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".
isTamperDetectable	xsd:NMTOKEN		none	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".

Source	<xsd:element name="DocumentEnvelope">

```
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="Attachment" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attributeGroup ref="name"/>
   <xsd:attribute name="businessDocument" type="xsd:string" use="required"/>
   <xsd:attribute name="businessDocumentIDRef" type="xsd:IDREF"/>
   <xsd:attribute name="isPositiveResponse" type="xsd:boolean"/>
   <xsd:attributeGroup ref="documentSecurity"/>
  </xsd:complexType>
</xsd:element>
```

8.1.12 Element: DocumentSubstitution

diagram



namespace: http://www.ebxml.org/BusinessProcess

Description: DocumentSubstitution specifies a document that should be used in place of a document in an existing process specification.

children: **Documentation**

used by: element **SubstitutionSet**

attributes

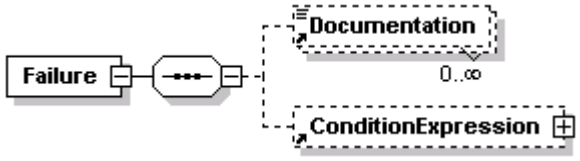
Name	Type	Use	Default	Annotation
originalBusinessDocument	xsd:string	required		The name of a business document within the scope of the substitution set.
originalBusinessDocumentID	xsd:ID			The ID of the business document.
substituteBusinessDocumentLocation	xsd:anyURI	required		The location of the document which shall replace the current document.
substituteBusinessDocumentId	xsd:anyURI			The ID of the replacement document.

source

```
<xsd:element name="DocumentSubstitution">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="originalBusinessDocument" type="xsd:string" use="required"/>
  <xsd:attribute name="originalBusinessDocumentID" type="xsd:ID"/>
  <xsd:attribute name="substituteBusinessDocumentLocation" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="substituteBusinessDocumentId" type="xsd:anyURI"/>
 </xsd:complexType>
</xsd:element>
```

8.1.13 Element: Failure

Diagram



Namespace: http://www.ebxml.org/BusinessProcess

Description: Defines the unsuccessful conclusion of a binary collaboration as a transition from an activity.

Wellformedness Rules: Every Binary Collaboration should have at least one failure.

Children	**Documentation** **ConditionExpression**
Used by	element **BinaryCollaboration**

Attributes	Name	Type	Use	Default	Annotation
	nameID	xsd:ID			Defines ID of the Failure
	fromBusinessState	xsd:string	required		The name of the activity from which this indicates a transition to unsuccessful conclusion of the BusinessTransaction or BinaryCollaboration
	fromBusinessStateIDRef	xsd:IDREF			The XML IDREF version of fromBusinessState
	conditionGuard	xsd:NMTOKEN			The condition that guards this transition

Source

```
<xsd:element name="Failure">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ConditionExpression" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="nameID" type="xsd:ID"/>
  <xsd:attribute name="fromBusinessState" type="xsd:string" use="required"/>
  <xsd:attribute name="fromBusinessStateIDRef" type="xsd:IDREF"/>
  <xsd:attribute name="conditionGuard">
   <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
     <xsd:enumeration value="ProtocolSuccess"/>
     <xsd:enumeration value="AnyProtocolFailure"/>
     <xsd:enumeration value="RequestReceiptFailure"/>
     <xsd:enumeration value="RequestAcceptanceFailure"/>
     <xsd:enumeration value="ResponseReceiptFailure"/>
     <xsd:enumeration value="ResponseAcceptanceFailure"/>
     <xsd:enumeration value="SignalTimeout"/>
     <xsd:enumeration value="ResponseTimeout"/>
     <xsd:enumeration value="BusinessSuccess"/>
     <xsd:enumeration value="BusinessFailure"/>
     <xsd:enumeration value="Success"/>
     <xsd:enumeration value="Failure"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:attribute>
 </xsd:complexType>
</xsd:element>
```

8.1.14 Element: Fork

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	A Fork is a state with one inbound transition and multiple outbound transitions. All activities pointed to by the outbound transitions are assumed to happen in parallel or exclusive or.
Children	**Documentation**
Used by	element **BinaryCollaboration**

Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Fork state
	nameID	xsd:ID			The XML ID version of name
	type	xsd:NMTOKEN	required	All	All : all activities will run in parallel. XOR:

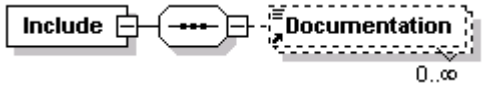
	timeToPerform xsd:duration optional	Only one of the possible activities will run. timeToPerform attribute on the Fork element may be used to specify that the business activities between the Fork and the Join shall be executed within the specified duration otherwise, the state of the collaboration will automatically advance to the join.

Source

```
<xsd:element name="Fork">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="type" use="optional" default="All">
   <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
     <xsd:enumeration value="All"/>
     <xsd:enumeration value="XOR"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="timeToPerform" type="xsd:duration" use="optional"/>
 </xsd:complexType>
</xsd:element>
```

8.1.15 Element: Include

Diagram



Namespace: http://www.ebxml.org/BusinessProcess

Description: Includes another process specification document and merges that specification with the current specification. Any elements of the same name and in the same name scope must have exactly the same specification except that packages may have additional content.Documents are merged based on name scope. A name in an included package will be indistinguishable from a name in the base document.

Children: **Documentation**

Used by: elements **Package** **ProcessSpecification**

Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Name of the included specification
uuid	xsd:ID	required		Unique identifier of the included specification
uri	xsd:anyURI	required		URI of the included specification
version	xsd:string	required		Version of the included specification

Source

```
<xsd:element name="Include">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="uuid" type="xsd:ID" use="required"/>
  <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="version" type="xsd:string" use="required"/>
 </xsd:complexType>
</xsd:element>
```

8.1.16 Element:Role

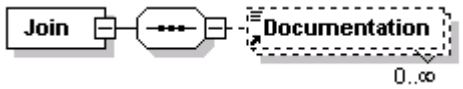
Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
type	**RoleType**
Children	**Documentation**
Used by	element **BinaryCollaboration**

Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Initiating Role
	nameID	xsd:ID			XML ID version the name.

Source

```
<xsd:element name="Role" type="RoleType" minOccurs="2" maxOccurs="2"/>
```

8.1.17 Element: Join

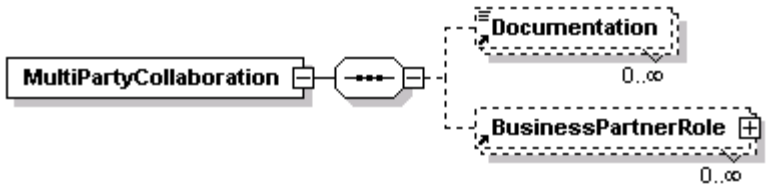
Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	A business state where an activity is waiting for the completion of one or more other activities. Defines the point where previously forked activities join up again.
Children	**Documentation**
Used by	element **BinaryCollaboration**

Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Join state.
	nameID	xsd:ID			The XML ID version of name
	waitForAll	xsd:boolean		true	Boolean value indicating if this Join state should wait for all incoming transitions to complete. If TRUE, wait for all, if False proceed on first incoming transition.

Source

```
<xsd:element name="Join">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="name"/>
    <xsd:attribute name="waitForAll" type="xsd:boolean" default="true"/>
  </xsd:complexType>
</xsd:element>
```

8.1.18 Element: MultiPartyCollaboration

diagram	
namespace	http://www.ebxml.org/BusinessProcess
Description	A Multiparty Collaboration is a synthesis of Binary Collaborations. A Multiparty Collaboration consists of a number of Business Partner Roles each playing roles in binary collaborations with each other. Wellformedness Rules: All multiparty collaborations must be synthesized from binary collaborations
children	**Documentation** **BusinessPartnerRole**
used by	elements **Package** **ProcessSpecification**

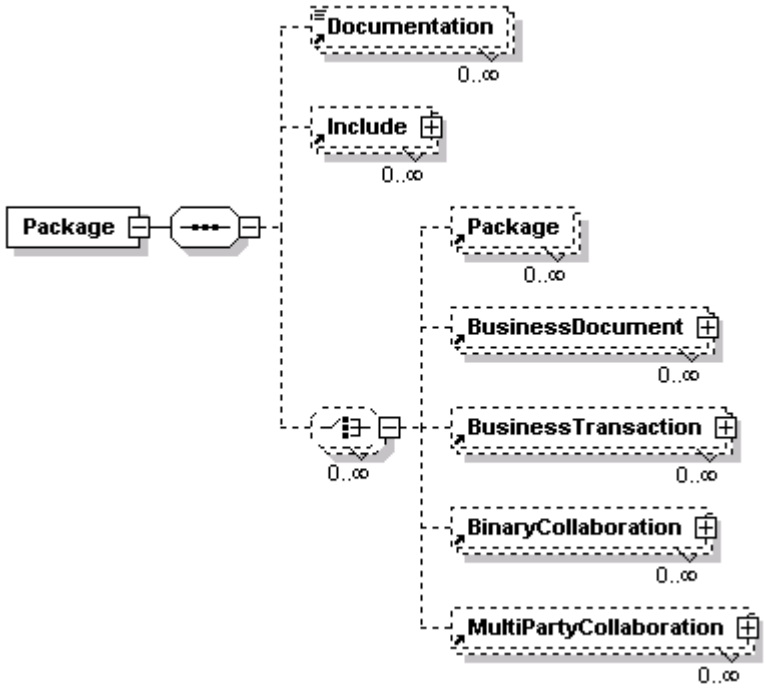
attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the MultiPartyCollaboration
nameID	xsd:ID			The XML ID version of name

source

```
<xsd:element name="MultiPartyCollaboration">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="BusinessPartnerRole" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
 </xsd:complexType>
</xsd:element>
```

8.1.19 Element: Package

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Children	**Documentation Include Package BusinessDocument BusinessTransaction BinaryCollaboration MultiPartyCollaboration**
Description	Defines a hierarchical name scope containing reusable elements.
Used by	elements **Package ProcessSpecification**

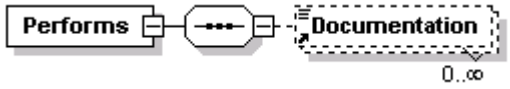
Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Name of the package
nameID	xsd:ID			XML ID version of name

Source

```
<xsd:element name="Package">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Include" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="Package" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BusinessDocument" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BusinessTransaction" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BinaryCollaboration" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="MultiPartyCollaboration" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:choice>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
 </xsd:complexType>
</xsd:element>
```

8.1.20 Element: Performs

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	Performs is an explicit modeling of the relationship between a BusinessPartnerRole and the Roles it plays. This specifies the use of an Authorized Role within a multiparty collaboration. Wellformedness Rules: For every Performs performing an AuthorizedRole there must be a Performs that performs the opposing AuthorizedRole, otherwise the MultiParty Collaboration is not complete.
Children	**Documentation**
Used by	element **BusinessPartnerRole**

Attributes

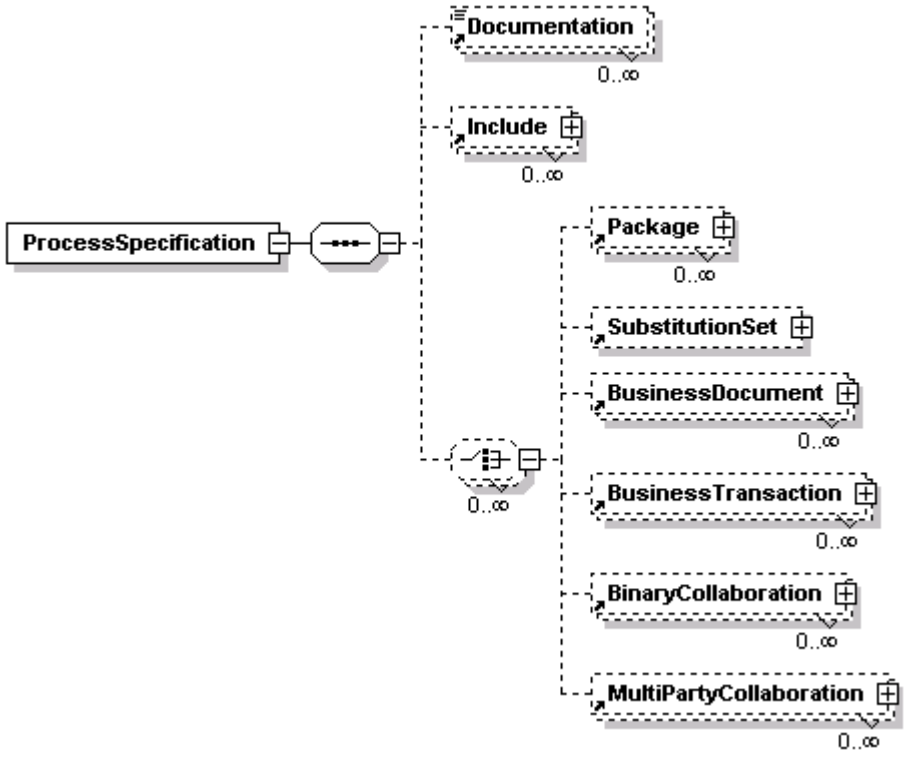
Name	Type	Use	Default	Annotation
nameID	xsd:ID			Defines ID of the DocumentEnvelope
role	xsd:string	optional		The Role that will be performed by the Business PartnerRole, qualified with the name of the BinaryCollaboration
roleIDRef	xsd:IDREF	optional		The XML IDREF version of Role

Source

```
<xsd:element name="Performs">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="nameID" type="xsd:ID"/>
  <xsd:attribute name="role" type="xsd:string" use="required"/>
  <xsd:attribute name="roleIDRef" type="xsd:IDREF" use="required"/>
 </xsd:complexType>
</xsd:element>
```

8.1.21 Element: ProcessSpecification

Diagram



Namespace	http://www.ebxml.org/BusinessProcess
Description	Root element of a process specification document that has a globally unique identity.
children	**Documentation Include Package SubstitutionSet BusinessDocument BusinessTransaction BinaryCollaboration MultiPartyCollaboration**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the ProcessSpecification element.
uuid	xsd:anyURI	required		The ID of the ProcessSpecification element.
version	xsd:string	required		Version of the specification.

identity constraints

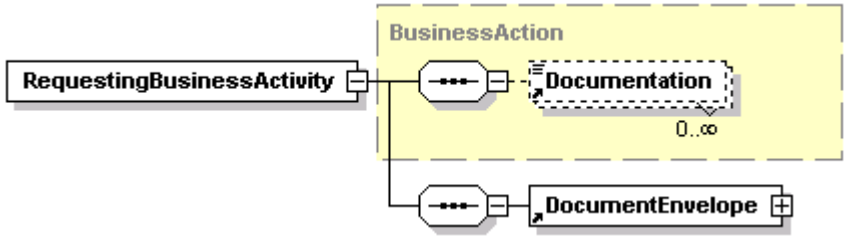
	Name	Refer	Selector	Field(s)
unique	ProcessSpecification-ID		.	uuid

source

```
<xsd:element name="ProcessSpecification">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Include" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="Package" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="SubstitutionSet" minOccurs="0"/>
    <xsd:element ref="BusinessDocument" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BusinessTransaction" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BinaryCollaboration" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="MultiPartyCollaboration" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="uuid" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="version" type="xsd:string" use="required"/>
 </xsd:complexType>
```

```
  <xsd:unique name="ProcessSpecification-ID">
    <xsd:selector xpath="."/>
    <xsd:field xpath="uuid"/>
  </xsd:unique>
</xsd:element>
```

8.1.22 Element: RequestingBusinessActivity

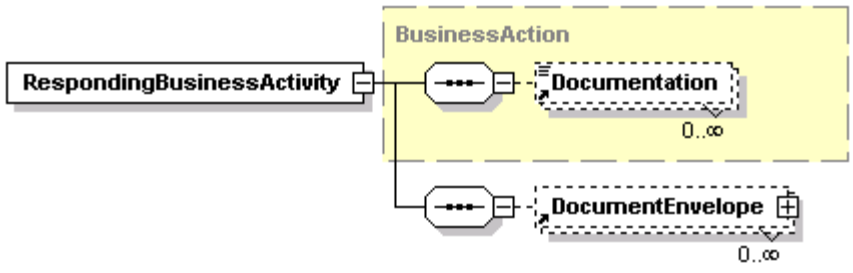
Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	A RequestingBusinessActivity is a Business Action that is performed by the requesting role within a Business Transaction. It specifies the Document Envelope which will carry the request.
type	extension of **BusinessAction**
Children	**Documentation** **DocumentEnvelope**
Used by	element **BusinessTransaction**

Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the RequestingBusinessTransaction
nameID	xsd:ID			The XML ID version of name
isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originator against a list of authorized originators.This parameter is specified on the sending side. (See also section on action security)
isIntelligibleCheckRequired	xsd:boolean		false	Receiving party must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt This parameter is specified on the sending side.(See also section on core transaction semantics)
isNonRepudiationReceiptRequired	xsd:boolean		false	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt.This parameter is specified on the sending side.(See also section on nonrepudiation)

	retryCount xsd:int	Business Level Retries - Aligns with UMM. The message id be different when a retried from business process level and the receiving application will be responsible for detecting for business level duplicates using the unique id((like a purchase order number)) in the business information.
Source	<xsd:element name="RequestingBusinessActivity"> <xsd:complexType> <xsd:complexContent> <xsd:extension base="BusinessAction"> <xsd:sequence> <xsd:element ref="DocumentEnvelope"/> </xsd:sequence> <xsd:attribute name="retryCount" type="xsd:int"/> </xsd:extension> </xsd:complexContent> </xsd:complexType> </xsd:element>	

8.1.23 Element: RespondingBusinessActivity

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	A RespondingBusinessActivity is a Business Action that is performed by the responding role within a Business Transaction. It specifies the Document Envelope which will carry the response.There may be multiple possible response Document Envelopes defined, but only one of them will be sent during an actual transaction instance.
type	extension of **BusinessAction**

Children	**Documentation** **DocumentEnvelope**
Used by	element **BusinessTransaction**

Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the RespondingBusinessTransaction
nameID	xsd:ID			The XML ID version of name
isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originator against a list of authorized originators.This parameter is specified on the sending side. (See also section on action security)
isIntelligibleCheckRequired	xsd:boolean		false	Receiving party must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt This parameter is specified on the sending side.(See also section on core transaction semantics)
isNonRepudiationReceiptRequired	xsd:boolean		false	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt.This parameter is specified on the sending side.(See also section on nonrepuditation)
isNonRepudiationRequired	xsd:boolean		false	Requires the sending parties to save copies of the transacted documents before sending them(See also section on nonrepuditation)
timeToAcknowledgeReceipt	xsd:duration			The time the receiving party has to acknowledge receipt of a business document. This parameter is specified on the sending side.(See also section on core transaction semantics)
timeToAcknowledgeAcceptance	xsd:duration			The time a responding role has to non-substantively acknowledge business acceptance of a business document.This parameter is specified on the requesting side.(See also section on core transaction semantics)

Source

```
<xsd:element name="RespondingBusinessActivity">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="BusinessAction">
        <xsd:sequence>
          <xsd:element ref="DocumentEnvelope" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

8.1.24 Element: Start

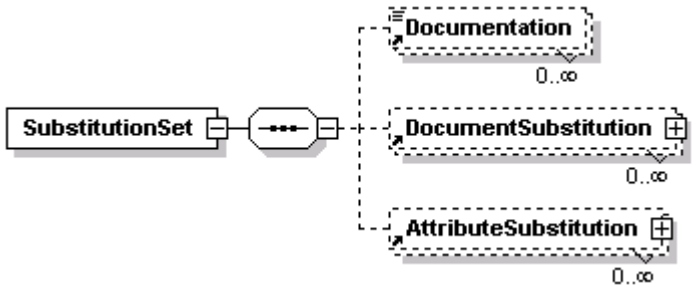
Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	The starting state for an Binary Collaboration. A Binary Collaboration should have only one starting activity.
Children	**Documentation**
Used by	element **BinaryCollaboration**

Attributes	Name	Type	Use	Default	Annotation
	toBusinessState	xsd:string	required		The name of an activity which an allowable starting point for this for BinaryCollaboration
	toBusinessState IDRef	xsd:IDREF			The XML IDREF version of toBusinessState

Source

```
<xsd:element name="Start">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="toBusinessState" type="xsd:string" use="required"/>
  <xsd:attribute name="toBusinessStateIDRef" type="xsd:IDREF"/>
 </xsd:complexType>
</xsd:element>
```

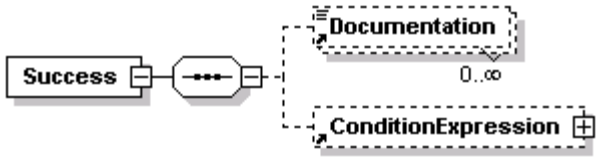
8.1.25 Element: SubstitutionSet

diagram	
namespace	http://www.ebxml.org/BusinessProcess
Description	A Substitution Set is a container for one or more AttributeSubstitution and/or DocumentSubstitution elements. The entire SubstitutionSet specifies document or attribute values that should be used in place of some documents and attribute values in an existing process specification.
children	**Documentation** **DocumentSubstitution** **AttributeSubstitution**
used by	element **ProcessSpecification**

attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Name of the substitution set.
	nameId	xsd:ID	required		The ID of the substitution set.
	applyToScope	xsd:string	required		Specifies the path to attributes or documents that are to be substituted for.

source	<xsd:element name="SubstitutionSet"> <xsd:complexType> <xsd:sequence> <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/> <xsd:element ref="DocumentSubstitution" minOccurs="0" maxOccurs="unbounded"/> <xsd:element ref="AttributeSubstitution" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> <xsd:attributeGroup ref="name"/> <xsd:attribute name="applyToScope" type="xsd:string" use="required"/> </xsd:complexType> </xsd:element>

8.1.26 Element: Success

Diagram	
Namespace	http://www.ebxml.org/BusinessProcess
Description	Defines the successful conclusion of a binary collaboration as a transition from an activity. Wellformedness Rules: Every activity Binary Collaboration should have at least one success.
Children	**Documentation** **ConditionExpression**
Used by	element **BinaryCollaboration**

Attributes

Name	Type	Use	Default	Annotation
nameID	xsd:ID			Defines ID of the DocumentEnvelope
fromBusinessState	xsd:string	required		The name of the activity from which this indicates a transition to successful conclusion of the BusinessTransaction or BinaryCollaboration
fromBusinessStateIDRef	xsd:IDREF			ID of the business state
conditionGuard	xsd:NMTOKEN			The condition that guards this transition

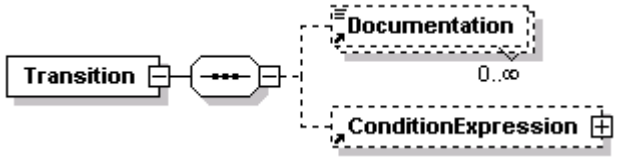
source

```
<xsd:element name="Success">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ConditionExpression" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="nameID" type="xsd:ID"/>
  <xsd:attribute name="fromBusinessState" type="xsd:string" use="required"/>
  <xsd:attribute name="fromBusinessStateIDRef" type="xsd:IDREF"/>
  <xsd:attribute name="conditionGuard">
   <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
     <xsd:enumeration value="ProtocolSuccess"/>
     <xsd:enumeration value="AnyProtocolFailure"/>
     <xsd:enumeration value="RequestReceiptFailure"/>
     <xsd:enumeration value="RequestAcceptanceFailure"/>
     <xsd:enumeration value="ResponseReceiptFailure"/>
     <xsd:enumeration value="ResponseAcceptanceFailure"/>
     <xsd:enumeration value="SignalTimeout"/>
     <xsd:enumeration value="ResponseTimeout"/>
     <xsd:enumeration value="BusinessSuccess"/>
     <xsd:enumeration value="BusinessFailure"/>
     <xsd:enumeration value="Success"/>
     <xsd:enumeration value="Failure"/>
```

```
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
 </xsd:complexType>
</xsd:element>
```

8.1.27 Element: Transition

Diagram



Namespace: http://www.ebxml.org/BusinessProcess

Description: A transition is a transition between two business states in a binary collaboration. Choreography is expressed as transitions between business states.Transition to the same state is allowed.

Children: **Documentation** **ConditionExpression**

Used by: elements **BinaryCollaboration** **BusinessPartnerRole**

Attributes

Name	Type	Use	Default	Annotation
nameID	xsd:ID			Defines ID of the DocumentEnvelope
onInitiation	xsd:boolean		false	This specifies this is a nested BusinessTransactionActivity and that upon receipt of the request in the associated transaction a second activity is performed before returning to the transaction to send the response back to the original requestor
fromBusinessState	xsd:string	required		The name of the state transitioned from.
fromBusinessStateIDRef	xsd:IDREF	optional		The XML IDREF version of fromBusinessState
toBusinessState	xsd:string	required		The name of the state transitioned to
toBusinessStateIDRef	xsd:IDREF	optional		The XML IDREF version of toBusinessState
conditionGuard	xsd:NMTOKEN			A reference to the status of the previous transaction. A fixed value of Success, BusinessFailure, TechnicalFailure, or AnyFailure

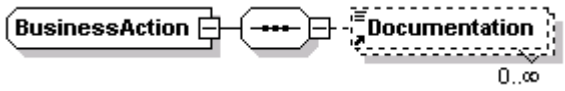
source

```
<xsd:element name="Transition">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ConditionExpression" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="nameID" type="xsd:ID"/>
  <xsd:attribute name="onInitiation" type="xsd:boolean" default="false"/>
  <xsd:attribute name="fromBusinessState" type="xsd:string" use="required"/>
  <xsd:attribute name="fromBusinessStateIDRef" type="xsd:IDREF" use="optional"/>
  <xsd:attribute name="toBusinessState" type="xsd:string" use="required"/>
  <xsd:attribute name="toBusinessStateIDRef" type="xsd:IDREF" use="optional"/>
  <xsd:attribute name="conditionGuard">
   <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
     <xsd:enumeration value="ProtocolSuccess"/>
     <xsd:enumeration value="AnyProtocolFailure"/>
     <xsd:enumeration value="RequestReceiptFailure"/>
     <xsd:enumeration value="RequestAcceptanceFailure"/>
     <xsd:enumeration value="ResponseReceiptFailure"/>
     <xsd:enumeration value="ResponseAcceptanceFailure"/>
     <xsd:enumeration value="SignalTimeout"/>
     <xsd:enumeration value="ResponseTimeout"/>
```

```
        <xsd:enumeration value="BusinessSuccess"/>
        <xsd:enumeration value="BusinessFailure"/>
        <xsd:enumeration value="Success"/>
        <xsd:enumeration value="Failure"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
 </xsd:complexType>
</xsd:element>
```

8.1.28 complexType BusinessAction

diagram



namespace: http://www.ebxml.org/BusinessProcess

description:

children: **Documentation**

used by: elements **RequestingBusinessActivity** **RespondingBusinessActivity**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the RequestingBusinessTransaction or RespondingBusinessTransaction depending on the subtype
nameID	xsd:ID			Defines the nameID
isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originator against a list of authorized originators. This parameter is specified on the sending side.
isIntelligibleCheckRequired	xsd:boolean		false	Receiving party must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt. This parameter is specified on the sending side.
isNonRepudiationReceiptRequired	xsd:boolean		false	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt. This parameter is specified on the sending side. (See also section on nonrepuditation)
isNonRepudiationRequired	xsd:boolean		false	Requires the sending parties to save copies of the transacted documents before sending them(See also section on nonrepuditation)
timeToAcknowledgeReceipt	xsd:duration			The time a receiving role has to acknowledge receipt of a business document. This parameter is specified on the sending side.
timeToAcknowledgeAcceptance	xsd:duration			The time a responding role has to non-substantively acknowledge business acceptance of a business document.This parameter is specified on the requesting side.(See also section on core transaction semantics)

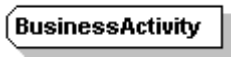
source

```
<xsd:complexType name="BusinessAction">
 <xsd:sequence>
  <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
 <xsd:attributeGroup ref="name"/>
```

```
  <xsd:attribute name="isAuthorizationRequired" type="xsd:boolean" default="false"/>
  <xsd:attribute name="isIntelligibleCheckRequired" type="xsd:boolean" default="false"/>
  <xsd:attribute name="isNonRepudiationRequired" type="xsd:boolean" default="false"/>
  <xsd:attribute name="isNonRepudiationReceiptRequired" type="xsd:boolean" default="false"/>
  <xsd:attribute name="timeToAcknowledgeReceipt" type="xsd:duration"/>
  <xsd:attribute name="timeToAcknowledgeAcceptance" type="xsd:duration"/>
</xsd:complexType>
```

8.1.29 complexType BusinessActivity

diagram



namespace: http://www.ebxml.org/BusinessProcess

used by: elements **BusinessTransactionActivity** **CollaborationActivity**

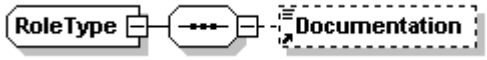
attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the activity uniquely within the binary collaboration
nameID	xsd:ID			The XML ID version of name
fromRole	xsd:string	required		The name of the initiating role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the requestor in the BusinessTransaction performed by this activity
fromRoleIDRef	xsd:IDREF			The XML IDREF version of fromAuthorizedRole
toRole	xsd:string	required		The name of the responding role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the responder in the BusinessTransaction performed by this activity
toRoleIDRef	xsd:IDREF			The XML IDREF version of toAuthorizedRole
beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.
postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration.

source

```
<xsd:complexType name="BusinessActivity">
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="fromRole" type="xsd:string" use="required"/>
  <xsd:attribute name="fromRoleIDRef" type="xsd:IDREF"/>
  <xsd:attribute name="toRole" type="xsd:string" use="required"/>
  <xsd:attribute name="toRoleIDRef" type="xsd:IDREF"/>
  <xsd:attribute name="beginsWhen" type="xsd:string"/>
  <xsd:attribute name="endsWhen" type="xsd:string"/>
  <xsd:attribute name="preCondition" type="xsd:string"/>
  <xsd:attribute name="postCondition" type="xsd:string"/>
</xsd:complexType>
```

8.1.30 complexType RoleType

diagram	
namespace	http://www.ebxml.org/BusinessProcess
children	**Documentation**
used by	elements **Role**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	xsd:string	required			
nameID	xsd:ID				

source

```
<xsd:complexType name="RoleType">
 <xsd:sequence>
  <xsd:element ref="Documentation" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="nameID" type="xsd:ID" use="required"/>
</xsd:complexType>
```

8.1.31 attributeGroup documentSecurity

namespace	http://www.ebxml.org/BusinessProcess
used by	elements **Attachment DocumentEnvelope**

attributes

Name	Type	Use	Default	Annotation
isAuthenticated	xsd:NMTOKEN		none	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.
isConfidential	xsd:NMTOKEN		none	The information entity is encrypted so that unauthorized parties cannot view the information.
isTamperDetectable	xsd:NMTOKEN		none	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.

source

```
<xsd:attributeGroup name="documentSecurity">
 <xsd:attribute name="isAuthenticated" default="none">
  <xsd:simpleType>
   <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="transient"/>
    <xsd:enumeration value="persistent"/>
    <xsd:enumeration value="transient-and-persistent"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:attribute>
 <xsd:attribute name="isConfidential" default="none">
  <xsd:simpleType>
   <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="transient"/>
    <xsd:enumeration value="persistent"/>
    <xsd:enumeration value="transient-and-persistent"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:attribute>
 <xsd:attribute name="isTamperProof" default="none">
  <xsd:simpleType>
```

```
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="none"/>
        <xsd:enumeration value="transient"/>
        <xsd:enumeration value="persistent"/>
        <xsd:enumeration value="transient-and-persistent "/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>
```

8.1.32 attributeGroup name

namespace	http://www.ebxml.org/BusinessProcess
used by	elements: Attachment BinaryCollaboration BusinessDocument BusinessPartnerRole BusinessTransaction BusinessTransactionActivity CollaborationActivity DocumentEnvelope Fork Join MultiPartyCollaboration Package SubstitutionSet complexTypes: BusinessAction AuthorizedRole
attributes	Name: name, nameID; Type: xsd:string, xsd:ID; Use: required; Default: ; Annotation:
source	`<xsd:attributeGroup name="name"><xsd:attribute name="name" type="xsd:string" use="required"/><xsd:attribute name="nameID" type="xsd:ID"/></xsd:attributeGroup>`

8.2 XML to UML cross-reference

The following is a table that references the XML element names in the XSD to their counterpart classes in the UML specification schema.

XML Element	UML Class
Attachment	Attachment
Role	AuthorizedRole
Binary Collaboration	Binary Collaboration
BusinessPartner Role	BusinessPartner Role
Business Transaction Activity	Business Transaction Activity
Business Transaction	Business Transaction
Responding BusinessActivity	Responding BusinessActivity
Requesting	Requesting

BusinessActivity	BusinessActivity
Collaboration Activity	Collaboration Activity
DocumentEnvelop e	DocumentEnvelope
Documentation	None (Should be added)
ebXML Process Specification	(From Package model: ebXML Process Specification)
Failure	Failure
Include	(From Package model: Include)
MultiParty Collaboration	MultiParty Collaboration
Package	(From Package model: Package)
Performs	Performs
Schema	Schema
Fork	Fork
Start	Start
Success	Success
Join	Join
Transition	Transition
BusinessAction	BusinessAction
DocumentSecurity	DocumentSecurity

The following classes in the UML specification schema are abstract, and do not have an element equivalent in theSchema. Only their concrete subtypes are in the Schema

- BusinessState
- CompletionState
- BusinessActivity

8.3 Scoped Name Reference

The structure of ebXML process specifications encourages re-use. An ebXMLProcessSpecification can include another ebXMLProcessSpecification by reference.

In addition the contents of a ProcessSpecification can be arranged in a recursive package structure. The ProcessSpecification is a package container, so it can contain packages within it. Package in itself is also a package container, so it can contain further packages within it.

Packages function as namespaces as per below.

Finally a Package, at any level can have PackageContent. Types of Package Content are BusinessDocument, BusinessTransaction, BinaryCollaboration, MultiPartyCollaboration.

Package Content are always uniquely named within a package. Lower level elements are uniquely named within their parent PackageContent.

Each Package Content type is a built-in context provider for the core components Logical Model for the Business Document definitions referenced by this ebXMLProcessSpecification.

Within a ebXMLProcessSpecification the following applies to naming:

Specification elements reference other specification elements by name through the use of attributes. The design pattern is that elements have a name attribute and other elements that reference the named elements do so through an attribute defined as the lowerCamelCase version of the referenced element (e.g. AuthorizedRole has attribute name while Performs, which references AuthorizedRole, has attribute authorizedRole). Two types of attributes are provided for names and references, XML ID/IDREF based and plain text. Each named element has a required name attribute and an optional nameID attribute. Referencing elements have lowerCamelCase and lowerCamelCaseIDRef attributes for the referenced element. XML ID/IDREF functionality requires all IDs to be unique within a document and that all IDREFs point to a defined ID value. Plain text attributes do not have this capability and may result in duplicate names. To unambiguously identify a referenced element using plain text attribute in the referencing attribute it is strongly recommended that XPath syntax be used. However, this is not enforced in the Schema.

The purpose of providing both solutions is to facilitate creation of Process Specification Documents directly in XML and to support future development tools that can automatically assign machine readable nameIDs and references. Both styles can be used simultaneously, in which case the ID and IDREF versions provide the unambiguous referencing and the plain text versions are used to provide meaningful names. Examples of named elements and references:

```
<Package name="ebXMLOrdering">
    <BinaryCollaboration name="OrderCollaboration" nameID="b112">
        <Role name="buyer" nameID="r224"/>
        <Role name="seller" nameID="r225"/>
    </BinaryCollaboration>
</Package>
```

<!-the XPath approach -->
<Performs
Role='//Package[@name="OAGOrdering"]/BinaryCollaboration[@name="OrderCollaboration"]/
Role[@name="buyer"]'/>

<!-Combination approach -->
<Performs Role="buyer" RoleIDRef="r224"/>

It is not required to use the full path specification as shown above, other forms of XPath expressions could be used as long as they resolve to a single reference. For example if buyer was unique to the document then the XPath could have been:

<Performs Role='//Role[@name="buyer"]'/>

Relative paths are also allowed for example:

<BusinessTransactionActivity fromRole='../ Role[@name="buyer"]' ... />

8.4 Sample XML document against above Schema

Provided in Appendix A

9 Business signal structures

The ebXML Message Service Specification signal structures provide business service state alignment infrastructure, including unique message identifiers and digests used to meet the basic process alignment requirements. The business signal payload structures provided herein are optional and normative and are intended to provide business and legal semantic to the business signals. Since signals do not differ in structure from business transaction to business transaction, they are defined once and for all, and their definition is implied by the conjunction of the Business Process Specification Schema and Message Service Specification. Here are the Schema's for business signal payload for ReceiptAcknowledgment and for AcceptanceAcknowledgement and Exception.

9.1.1 Signal Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
-->
<!-- By Himagiri Mukkamala(himagiri@sybase.com) -->
<xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified" version="2.0">
    <xsd:import namespace="http://www.w3.org/1999/xlink" schemaLocation="http://www.oasis-
open.org/committees/ebxml-msg/schema/xlink.xsd"/>
    <xsd:simpleType name="non-empty-string">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
        </xsd:restriction>
```

```
    </xsd:simpleType>
    <xsd:complexType name="PartyInfoType">
        <xsd:simpleContent>
            <xsd:extension base="non-empty-string">
                <xsd:attribute name="type" type="non-empty-string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="RoleType">
        <xsd:attribute name="name" type="non-empty-string" use="required"/>
        <xsd:attributeGroup ref="xlink.grp"/>
    </xsd:complexType>
    <xsd:attributeGroup name="xlink.grp">
        <xsd:attribute ref="xlink:type" fixed="simple"/>
        <xsd:attribute ref="xlink:href" use="required"/>
    </xsd:attributeGroup>
    <xsd:complexType name="ProcessSpecificationType">
        <xsd:attribute name="version" type="non-empty-string"/>
        <xsd:attribute name="name" type="non-empty-string"/>
        <xsd:attributeGroup ref="xlink.grp"/>
        <xsd:attribute name="uuid" type="xsd:anyURI"/>
    </xsd:complexType>
</xsd:schema>
```

9.1.2 ReceiptAcknowledgment Signal Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
-->
<!-- By Himagiri Mukkamala(himagiri@sybase.com) -->
<xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified" version="2.0">
    <xsd:include schemaLocation="Signal.xsd"/>
    <xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
    <xsd:element name="ReceiptAcknowledgment">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="OriginalMessageIdentifier" type="bpssignal:non-empty-string"/>
                <xsd:element name="OriginalDocumentIdentifier" type="bpssignal:non-empty-string"
minOccurs="0"/>
                <xsd:element ref="bpssignal:NonRepudiationInformation" minOccurs="0"/>
                <xsd:element name="FromPartyInfo" type="bpssignal:PartyInfoType"/>
                <xsd:element name="ToPartyInfo" type="bpssignal:PartyInfoType"/>
                <xsd:element name="FromRole" type="bpssignal:RoleType"/>
                <xsd:element name="ToRole" type="bpssignal:RoleType"/>
                <xsd:element name="OriginalMessageDateTime" type="xsd:dateTime"/>
                <xsd:element name="ThisMessageDateTime" type="xsd:dateTime"/>
                <xsd:element name="ProcessSpecificationInfo" type="bpssignal:ProcessSpecificationType"/>
                <xsd:element ref="ds:Signature" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="NonRepudiationInformation">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="bpssignal:MessagePartNRInformation" maxOccurs="unbounded"/>
            </xsd:sequence>
```

```
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="MessagePartNRInformation">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element name="MessagePartIdentifier" type="bpssignal:non-empty-string"/>
                <xsd:element ref="ds:Reference"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

9.1.3 AcceptanceAcknowledgement Signal Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
-->
<!-- By Himagiri Mukkamala(himagiri@sybase.com) -->
<xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified" version="2.0">
    <xsd:include schemaLocation="Signal.xsd"/>
    <xsd:element name="AcceptanceAcknowledgment">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="OriginalMessageIdentifier" type="bpssignal:non-empty-string"/>
                <xsd:element name="OriginalDocumentIdentifier" type="bpssignal:non-empty-string"
minOccurs="0"/>
                <xsd:element name="FromPartyInfo" type="bpssignal:PartyInfoType"/>
                <xsd:element name="ToPartyInfo" type="bpssignal:PartyInfoType"/>
                <xsd:element name="FromRole" type="bpssignal:RoleType"/>
                <xsd:element name="ToRole" type="bpssignal:RoleType"/>
                <xsd:element name="OriginalMessageDateTime" type="xsd:dateTime"/>
                <xsd:element name="ThisMessageDateTime" type="xsd:dateTime"/>
                <xsd:element name="ProcessSpecificationInfo" type="bpssignal:ProcessSpecificationType"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

9.1.4 Exception Signal Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
-->
<!--W3C Schema generated by XML Spy v4.2 U (http://www.xmlspy.com)-->
<xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified" version="2.0">
    <xsd:include schemaLocation="Signal.xsd"/>
    <xsd:element name="Exception">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="OriginalMessageIdentifier" type="bpssignal:non-empty-string"/>
```

```
                <xsd:element name="OriginalDocumentIdentifier" type="bpssignal:non-empty-string"
minOccurs="0"/>
                <xsd:element name="Reason" type="xsd:string"/>
                <xsd:element name="FromPartyInfo" type="bpssignal:PartyInfoType"/>
                <xsd:element name="ToPartyInfo" type="bpssignal:PartyInfoType"/>
                <xsd:element name="FromRole" type="bpssignal:RoleType"/>
                <xsd:element name="ToRole" type="bpssignal:RoleType"/>
                <xsd:element name="OriginalMessageDateTime" type="xsd:dateTime"/>
                <xsd:element name="ThisMessageDateTime" type="xsd:dateTime"/>
                <xsd:element name="ProcessSpecificationInfo" type="bpssignal:ProcessSpecificationType"/>
                <xsd:element name="ExceptionType">
                    <xsd:complexType>
                        <xsd:choice>
                            <xsd:element name="ReceiptException">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="Syntax"/>
                                        <xsd:enumeration value="Authorization"/>
                                        <xsd:enumeration value="Signature"/>
                                        <xsd:enumeration value="Sequence"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="AcceptanceException">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="Business"/>
                                        <xsd:enumeration value="Performance"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="GeneralException">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string"/>
                                </xsd:simpleType>
                            </xsd:element>
                        </xsd:choice>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="ExceptionMessage" type="xsd:string"/>
                <xsd:any namespace="##other" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

10 EDI support

A technical report will be made available to describe use of BPSS to describe EDI transactions.

11 Production Rules

This section provides a set of production rules, defining the mapping from the UML version of the *Business Process Specification Schema* to the XML version.

The primary purpose for these production rules is to govern the one-time generation of the Schemaversion of the *Business Process Specification Schema* from the UML Class Diagram version of *Business Process Specification Schema*.

The Class Diagram version of *Business Process Specification Schema* is not intended for the direct creation of ebXML Business Process Specifications. However, if a *Business Process Specification* was in fact (programmatically) created as an instance of this class diagram, the production rules would also provide the prescriptive definition necessary to translate a such an instance into a XML Specification Document conformant with theSchema. The production rules are defined for concrete classes, abstract classes, aggregate associations, specialization associations and unidirectional associations.

1. Classes are rendered as XML elements.
2. Class attributes are rendered as XML attributes. NOTE: occurrence requirements (required vs optional) and default values for attributes are not modeled.
3. Specialization classes (classes that inherit from another class) are rendered as XML elements including all attributes and aggregate associations from the base class. Repeated attributes are normalized to a single occurrence.
4. Abstract classes are not rendered in the XML Schema. Abstract classes are inherited from and represent a form of collection. A class that aggregates an abstract class, essentially aggregates “any of each” of the specialization classes.
5. An aggregate association renders the aggregated class as an XML child element with appropriate cardinality.
6. A unidirectional association defines an attribute in the originating class of the same name as the class the association points to. This type of attribute is called a “reference attribute” and contains the name of the class it points to. The referenced class must have a “name” attribute.
7. A class attribute data type, that has a class of the same name with stereotype <<Enumeration>> is rendered as an XML attribute enumeration. The Enumeration class does not have an explicit association.
8. A class attribute data type (e.g. Time, URI, Boolean) that has no corresponding class definition is rendered as a string in theSchema. In the XML Schema version these data types are mapped as:

   Time - xsd:duration
   URI - xsd:anyURI
   Boolean - xsd:boolean

9. Each class is given an optional “Documentation*” element which is intended for annotation of the specification instances. This is not modeled.

Appendix A: Sample XML Business Process Specification

```
<ProcessSpecification name="Simple" version="1.1" uuid="Simple-2434134"
xmlns="http://www.ebxml.org/BusinessProcess" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ebxml.org/BusinessProcess
ebBPSS1.04.xsd.xml">
    <!-- Business Documents -->
    <BusinessDocument name="Catalog Request" specificationLocation=”http://www.xyx.com/CatalogReq.xsd”/>
    <BusinessDocument name="Catalog" specificationLocation=”http://www.xyx.com/Catalog.xsd”/>
    <BusinessDocument name="Purchase Order" specificationLocation=”http://www.xyx.com/PO.xsd”/>
    <BusinessDocument name="PO Acknowledgement" specificationLocation=”http://www.xyx.com/POAck.xsd”/>
    <BusinessDocument name="Credit Request" specificationLocation=”http://www.xyx.com/CreditReq.xsd”/>
    <BusinessDocument name="Credit Confirm" specificationLocation=”http://www.xyx.com/CreditCon.xsd”/>
    <BusinessDocument name="ASN" specificationLocation=”http://www.xyx.com/CatalogASN.xsd”/>
    <BusinessDocument name="CreditAdvice" specificationLocation=”http://www.xyx.com/CreditAdvice.xsd”/>
    <BusinessDocument name="DebitAdvice" specificationLocation=”http://www.xyx.com/DebitAdvice.xsd”/>
    <BusinessDocument name="Invoice" specificationLocation=”http://www.xyx.com/Invoice.xsd”/>
    <BusinessDocument name="Payment" specificationLocation=”http://www.xyx.com/Payment.xsd”/>
    <BusinessDocument name="Inventory Report Request"
specificationLocation=”http://www.xyx.com/InvReq.xsd”/>
    <BusinessDocument name="Inventory Report" specificationLocation=”http://www.xyx.com/InvRep.xsd”/>
    <Package name="Ordering">

        <!-- Here are all the Business Transactions needed -->
        <BusinessTransaction name="Catalog Request">
            <RequestingBusinessActivity name="RequestCatalog">
                <DocumentEnvelope businessDocument="Catalog Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="SendCatalog">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="Catalog"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>

        <BusinessTransaction name="Create Order">
            <RequestingBusinessActivity name="SendOrder" isNonRepudiationRequired="true"
timeToAcknowledgeReceipt="P2D" timeToAcknowledgeAcceptance="P3D">
                <DocumentEnvelope businessDocument="Purchase Order"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="SendPOAcknowledgement" isNonRepudiationRequired="true"
timeToAcknowledgeReceipt="P5D">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="PO Acknowledgement"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>

        <BusinessTransaction name="Check Credit ">
            <RequestingBusinessActivity name="CreditCheck">
                <DocumentEnvelope businessDocument="Credit Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="ConfirmCredit">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="Credit Confirm"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>

        <BusinessTransaction name="Notify of advance shipment">
            <RequestingBusinessActivity name="AdvanceShipmentNotification">
                <DocumentEnvelope businessDocument="ASN"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="ASNResponse" />
        </BusinessTransaction>

        <BusinessTransaction name="Process Credit Payment">
```

```
            <RequestingBusinessActivity name="CreditPaymentProcess">
                <DocumentEnvelope businessDocument="CreditAdvice"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="CreditPaymentProcessResponse">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="DebitAdvice"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>

        <BusinessTransaction name="Process Payment">
            <RequestingBusinessActivity name="PaymentProcess">
                <DocumentEnvelope businessDocument="Invoice"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="SendPayment">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="Payment"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>

        <BusinessTransaction name="Request Inventory Report">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope businessDocument="Inventory Report Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="Inventory Report">
            <DocumentEnvelope businessDocument="Inventory Report"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>

        <!-- Now the Binary Collaborations -->
        <BinaryCollaboration name="Request Catalog" initiatingRoleID="1122B1”>
            <Role name="requestor" nameID="1122B1”/>
            <Role name="provider" nameID="2211A1”/>
            <Start toBusinessState="Catalog Request"/>
            <BusinessTransactionActivity name="Catalog Request" businessTransaction="Catalog Request"
fromRole="requestor" toRole="provider"/>
            <Success fromBusinessState="Catalog Request" conditionGuard="Success"/>
            <Failure fromBusinessState="Catalog Request" conditionGuard="Failure"/>
        </BinaryCollaboration>

        <BinaryCollaboration name="Firm Order" timeToPerform="P2D" initiatingRoleID="1122B2”>
            <Documentation>timeToPerform = Period: 2 days from start of transaction</Documentation>
            <Role name="buyer" nameID="1122B2”/>
            <Role name="seller" nameID="1122B3”/>
            <Start toBusinessState="Create Order"/>
            <BusinessTransactionActivity name="Create Order" businessTransaction="Create Order"
fromRole="buyer" toRole="seller"/>
            <Success fromBusinessState="Create Order" conditionGuard="Success"/>
            <Failure fromBusinessState="Create Order" conditionGuard="Failure"/>
        </BinaryCollaboration>

        <BinaryCollaboration name="Product Fulfillment" timeToPerform="P5D" initiatingRoleID="1122B2”>
            <Documentation>timeToPerform = Period: 5 days from start of transaction</Documentation>
            <Role name="buyer" nameID="1122B2”/>
            <Role name="seller" nameID="1122B3”/>
            <Start toBusinessState="Create Order"/>
            <BusinessTransactionActivity name="Create Order" businessTransaction="Create Order"
fromRole="buyer" toRole="seller"/>
            <BusinessTransactionActivity name="Notify shipment" businessTransaction="Notify of advance
shipment" fromRole="seller" toRole="buyer"/>
            <Transition fromBusinessState="Create Order" toBusinessState="Notify shipment"/>
            <Success fromBusinessState="Notify shipment" conditionGuard="Success"/>
            <Failure fromBusinessState="Notify shipment" conditionGuard="Failure"/>
        </BinaryCollaboration>

        <BinaryCollaboration name="Inventory Status" initiatingRoleID="1122B1”>
            <Role name="requestor" nameID="1122B1”/>
```

```
            <Role name="provider" nameID="2211A1"/>
            <Start toBusinessState="Inventory Report Request"/>
            <BusinessTransactionActivity name="Inventory Report Request" businessTransaction="Inventory
Report Request" fromRole="requestor" toRole="provider"/>
            <Success fromBusinessState=" Inventory Report Request " conditionGuard="Success"/>
            <Failure fromBusinessState=" Inventory Report Request " conditionGuard="Failure"/>
        </BinaryCollaboration>

        <BinaryCollaboration name="Credit Inquiry" initiatingRoleID="9122B1">
            <Role name="creditor"" nameID="9122B1"/>
            <Role name="credit service"" nameID="8122B1"/>
            <Start toBusinessState="Check Credit"/>
            <BusinessTransactionActivity name="Check Credit" businessTransaction="Check Credit"
fromRole="creditor" toRole="credit service"/>
            <Success fromBusinessState="Check Credit" conditionGuard="Success"/>
            <Failure fromBusinessState="Check Credit" conditionGuard="Failure"/>
        </BinaryCollaboration>

        <BinaryCollaboration name="Credit Payment" initiatingRoleID="6122B1">
            <Role name="payee" nameID="6122B1"/>
            <Role name="payor" nameID="7122B1"/>
            <Start toBusinessState="Process Credit Payment"/>
            <BusinessTransactionActivity name="Process Credit Payment" businessTransaction="Process Credit
Payment" fromRole="payee" toRole="payor"/>
            <Success fromBusinessState="Process Credit Payment" conditionGuard="Success"/>
            <Failure fromBusinessState="Process Credit Payment" conditionGuard="Failure"/>
        </BinaryCollaboration>

        <!-- A compound BinaryCollaboration for illustration purposes-->
        <BinaryCollaboration name="Credit Charge" initiatingRoleID="8132B1">
            <Role name="charger" nameID="8132B1"/>
            <Role name="credit service"" nameID="8122B1"/>
            <Start toBusinessState="Credit Inquiry"/>
            <CollaborationActivity name="Credit Inquiry" binaryCollaboration="Credit Inquiry" fromRole="charger"
toRole="credit service"/>
            <CollaborationActivity name="Credit Payment" binaryCollaboration="Credit Payment"
fromRole="charger" toRole="payor"/>
            <Transition fromBusinessState="Credit Inquiry" toBusinessState="Credit Payment"/>
            <Success fromBusinessState="Credit Payment" conditionGuard="Success"/>
            <Failure fromBusinessState="Credit Payment" conditionGuard="Failure"/>
        </BinaryCollaboration>

        <BinaryCollaboration name="Fulfillment Payment" initiatingRoleID="6122B1">
            <Role name="payee" nameID="6122B1"/>
            <Role name="payor" nameID="7122B1"/>
            <Start toBusinessState="Process Payment"/>
            <BusinessTransactionActivity name="Process Payment" businessTransaction="Process Payment"
fromRole="payee" toRole="payor"/>
            <Success fromBusinessState="Process Payment" conditionGuard="Success"/>
            <Failure fromBusinessState="Process Payment" conditionGuard="Failure"/>
        </BinaryCollaboration>

        <!-- First the overall MultiParty Collaboration -->
        <MultiPartyCollaboration name="DropShip">
            <BusinessPartnerRole name="Customer">
                <Performs role="requestor" roleIDRef="1122B1"/>
                <Performs role="buyer" roleIDRef="1122B2"/>
                <Transition fromBusinessState="Catalog Request" toBusinessState="Create Order"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="Retailer">
                <Performs role="provider" roleIDRef="2211A1"/>
                <Performs role="seller" roleIDRef="1122B3"/>
                <Performs role="creditor" roleIDRef="9122B1"/>
                <Performs role="buyer" roleIDRef="1122B2"/>
```

```
                <Performs role="payee" roleIDRef="6122B1”/>
                <Performs role="payor" roleIDRef="7122B1”/>
                <Performs role="requestor" roleIDRef="1122B1”/>
                <Transition fromBusinessState="Create Order" toBusinessState="Check Credit"/>
                <Transition fromBusinessState="Check Credit" toBusinessState="Credit Payment"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="DropShip Vendor">
                <Performs role="seller" roleIDRef="1122B3”/>
                <Performs role="payee" roleIDRef="6122B1”/>
                <Performs role="provider" roleIDRef="2211A1”/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="Credit Authority">
                <Performs role="credit service" roleIDRef="8122B1”/>
                <Performs role="payor" roleIDRef="7122B1”/>
            </BusinessPartnerRole>
        </MultiPartyCollaboration>
    </Package>
</ProcessSpecification>
```

Appendix B: Sample XML Signal

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
-->
<!--Sample XML file generated by XML Spy v4.2 U (http://www.xmlspy.com)-->
<bpssignal:Exception xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
xsd:schemaLocation="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS
ExceptionSignal.xsd">
    <bpssignal:OriginalMessageIdentifier>MessageIdentifier-1</bpssignal:OriginalMessageIdentifier>
    <bpssignal:Reason>State transition failure</bpssignal:Reason>
    <bpssignal:FromPartyInfo bpssignal:type="DUNS.com">PartyA</bpssignal:FromPartyInfo>
    <bpssignal:ToPartyInfo bpssignal:type="DUNS.com">PartyB</bpssignal:ToPartyInfo>
    <bpssignal:FromRole bpssignal:name="Buyer" xlink:type="simple"
xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
    <bpssignal:ToRole bpssignal:name="Seller" xlink:type="simple"
xlink:href="http://www.rosettanet.org/processes/3A4.xml#Seller"/>
    <bpssignal:OriginalMessageDateTime>2002-03-05T19:00:00</bpssignal:OriginalMessageDateTime>
    <bpssignal:ThisMessageDateTime>2002-03-05T20:00:00</bpssignal:ThisMessageDateTime>
    <bpssignal:ProcessSpecificationInfo bpssignal:version="2.0"
bpssignal:name="PIP3A4RequestPurchaseOrder" xlink:type="simple"
xlink:href="http://www.rosettanet.org/processes/3A4.xml" bpssignal:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
    <bpssignal:ExceptionType>
        <bpssignal:ReceiptException>Signature</bpssignal:ReceiptException>
    </bpssignal:ExceptionType>
    <bpssignal:ExceptionMessage>Signature Validation Failed for request
message</bpssignal:ExceptionMessage>
</bpssignal:Exception>
```

12 References

1. UN/CEFACT Modeling Methodology (UMM) as defined in the N090R10 specification, http://www.gefeg.com/tmwg/n090r10.htm
2. ebXML Technical Architecture Specification, version 1.04, http://www.ebxml.org/specs/ebTA.pdf
3. ebXML Core Components Dictionary, version 1.04, http://www.ebxml.org/specs/ccDICT.pdf
4. ebXML Naming Convention for Core Components, version 1.04, http://www.ebxml.org/specs/ebCCNAM.pdf
5. ebXML Business Process and Business Information Analysis Overview, version 1.0, http://www.ebxml.org/specs/bpOVER.pdf
6. ebXML Business Process Analysis Worksheets & Guidelines, version 1.0, http://www.ebxml.org/specs/bpWS.pdf
7. ebXML E-Commerce Patterns, version 1.0, http://www.ebxml.org/specs/bpPATT.pdf
8. ebXML Catalog of Common Business Processes, version 1.0, http://www.ebxml.org/specs/bpPROC.pdf
9. RosettaNet Implementation Framework: Core Specification, Version: Release 2.00.00, 13 July 2001
10. Key Words for use in RFCs to Indicate Requirement Levels, Internet Engineering Task Force RFC 2119, http://www.ietf.org/rfc/rfc2119.txt.
11. Extensible Markup Language (XML), World Wide Web Consortium, http://www.w3.org/XML.
12. XML Schema Part 1: Structures, Worldwide Web Consortium, http://www.w3.org/TR/xmlschema-1/.
13. XML Schema Part 2: Datatypes, Worldwide Web Consortium,http://www.w3.org/TR/xmlschema-2/.
14. ebXML Message Service Specification, http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.
15. ebXML Registry Services Specification, http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf.
16. ebXML Collaboration-Protocol Profile and Agreement Specification V1.9, http://www.oasis-open.org/committees/ebxml-cppa/documents/working_drafts/ebCPP-1_9.pdf

13 Disclaimer

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

14 Contact Information

Team Leader (Of the BP team):
Brian Hayes
Collaborative Domain
Tel: (925) 788-6303
mailto:brian.hayes@UCLAlumni.net

Editor (of this document):

Pallavi Malu (Project Editor)
Intel Corporation
5000 W.Chandler Blvd,
Chandler, AZ 85226
Tel: 480-552-0463
mailto:pallavi.g.malu@intel.com

Jean-Jacques Dubray, (Editor – Sections 1-7)
Eigner *Precision Lifecycle Management*
200 Fifth Avenue
Waltham, MA 02451
Tel: 781-472-6317
jjd@eigner.com