ebXML Business Process Specification Schema Version 1.0

Context/Metamodel Group
of the CC/BP Joint Delivery Team

27 April 2001

1 Status of this Document

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

This version:
EbXML_BPschema_1.0

***Latest version*:**
EbXML_BPschema_0.99

Previous version:
EbXML_BPschema_0.90

2 ebXML BP/CoreComponents metamodel participants

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

Team Lead:

Paul Levine, Telcordia

Editors:

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, Spolin Silverman & Cohen LLP: (Legal aspects)
Neal Smith, Chevron: (Issues Lists, and W3C schema)
John Yunker, Edifecs: (Signal structures)
Karsten Riemer, Sun Microsystems: (Overall Document)

Participants:

Antoine Lonjon, Mega
J.J. Dubray, Excelon
Bob Haugen, Logistical Software
Bill McCarthy, Michigan State University
Paul Levine, Telcordia
Brian Hayes, CommerceOne
Nita Sharma, Netfish
David Welsh, Nordstrom
Christopher Ferris, Sun Microsystems
Antonio Carrasco, Data Access Technologies

3 Table of Contents

1 Status of this Document i
2 ebXML BP/CoreComponents metamodel participants ii
3 Table of Contents iii
4 Introduction v
4.1 Summary of Contents of Document 1
4.2 Audience 1
4.3 Related Documents 1
4.4 Prerequisites 2
5 Design Objectives 2
5.1 Goals/Objectives/Requirements/Problem Description 2
5.2 Caveats and Assumptions 3
5.2.1 Relationship between *ebXML Business Process Specification Schema* and UMM 3
6 System Overview 5
6.1 Key Concepts of the ebXML Business Process Specification Schema 10
6.2 How to use the ebXML Business Process Specification Schema 14
6.3 How ebXML Business Process Specification Schema is used with other ebXML specifications 14
6.4 How to design collaborations and transactions, re-using at design time 16
6.4.1 Specify a Business Transaction and its Business Document Flow 16
6.4.2 Specify a Binary Collaboration 22
6.4.3 Specify a MultiParty Collaboration 25
6.4.4 Specify a Choreography 27
6.4.5 The whole model 31
6.5 Core Business Transaction Semantics 33
6.5.1 Interaction Predictability 33
6.5.2 Creating legally binding contracts 36
6.5.3 Non-Repudiation 37
6.5.4 Authorization security 38
6.5.5 Document security 39
6.5.6 Reliability 40
6.5.7 Parameters required for CPP/CPA 40
6.6 Run time Business Transaction semantics 40
6.6.1 Timeouts 41
6.6.2 Exceptions 43
6.7 Runtime Collaboration Semantics 46
6.8 Where the ebXML Business Process Specification Schema May Be Implemented 46
7 UML Element Specification 46
7.1 Business Collaborations 46
7.1.1 MultiPartyCollaboration 46
7.1.2 BusinessPartnerRole 47
7.1.3 Performs 47
7.1.4 AuthorizedRole 48
7.1.5 BinaryCollaboration 48

7.1.6 BusinessActivity 50
7.1.7 BusinessTransactionActivity 50
7.1.8 CollaborationActivity 51
7.2 Business Transactions 51
7.2.1 BusinessTransaction 51
7.2.2 Business Action 52
7.2.3 RequestingBusinessActivity 53
7.2.4 RespondingBusinessActivity 54
7.3 Document flow 55
7.3.1 Document Security 55
7.3.2 Document Envelope 55
7.3.3 BusinessDocument 56
7.3.4 DocumentSpecification 57
7.3.5 Attachment 57
7.4 Choreography within Collaborations. 58
7.4.1 BusinessState 58
7.4.2 Transition 59
7.4.3 Start 59
7.4.4 CompletionState 60
7.4.5 Success 60
7.4.6 Failure 61
7.4.7 Fork 61
7.4.8 Join 61
7.5 Definition and Scope 62
7.6 Collaboration and transaction well-formedness rules 62
8 ebXML Business Process Specification Schema – (DTD) 65
8.1 Documentation for the DTD 65
8.2 XML to UML cross-reference 92
8.3 Scoped Name Reference 93
8.4 Sample XML document against above DTD 95
9 Business signal structures 95
9.1.1 ReceiptAcknowledgment DTD 95
9.1.2 AcceptanceAcknowledgement DTD 97
9.1.3 Exception Signal DTD 99
10 Production Rules 101
Appendix A: Sample XML Business Process Specification 103
Appendix B: Business Process Specification Schema DTD 108
Appendix C: Business Process Specification Schema XML Schema 115
11 References 126
12 Disclaimer 126
13 Contact Information 127

4 Introduction

Executive Summary

The ebXML 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 N090R9.1 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).

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.

4.1 Summary of Contents of Document

This document describes the ebXML Specification Schema

This document describes the Specification Schema, both in its UML form and in its DTD form.

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 the 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].

4.2 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.

4.3 Related Documents

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

- ebXML Technical Architecture, version 1.0
- ebXML Core Components, version 1.0
- ebXML Naming Convention for Core Components, version 1.0
- ebXML Core Component and Business Process Document Overview, version 1.0
- ebXML Collaboration-Protocol Profile and Agreement Specification V0.95
- ebXML Business Process and Business Information Analysis Overview, version 0.7
- ebXML Business Process Analysis Worksheets & Guidelines, version 0.99
- ebXML E-Commerce Patterns, version 0.99
- ebXML Catalog of Common Business Processes, version 0.99

- ebXML Message Service Specification V0.99
- UN/CEFACT Modeling Methodology (UMM) as defined in the N090R9.1 specification

4.4 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)

5 Design Objectives

5.1 Goals/Objectives/Requirements/Problem Description

Business process models describe interoperable business processes that allow business partners to collaborate. Business process models for e-business must be turned into software components that collaborate on behalf of the business partners.

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

The ebXML 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 Business Process Specification.

The *ebXML Business Process Specification Schema* is available in two stand-alone representations, a UML version, and an XML version.

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 version 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. Both a DTD and a W3C Schema is provided.

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

5.2 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.

5.2.1 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.

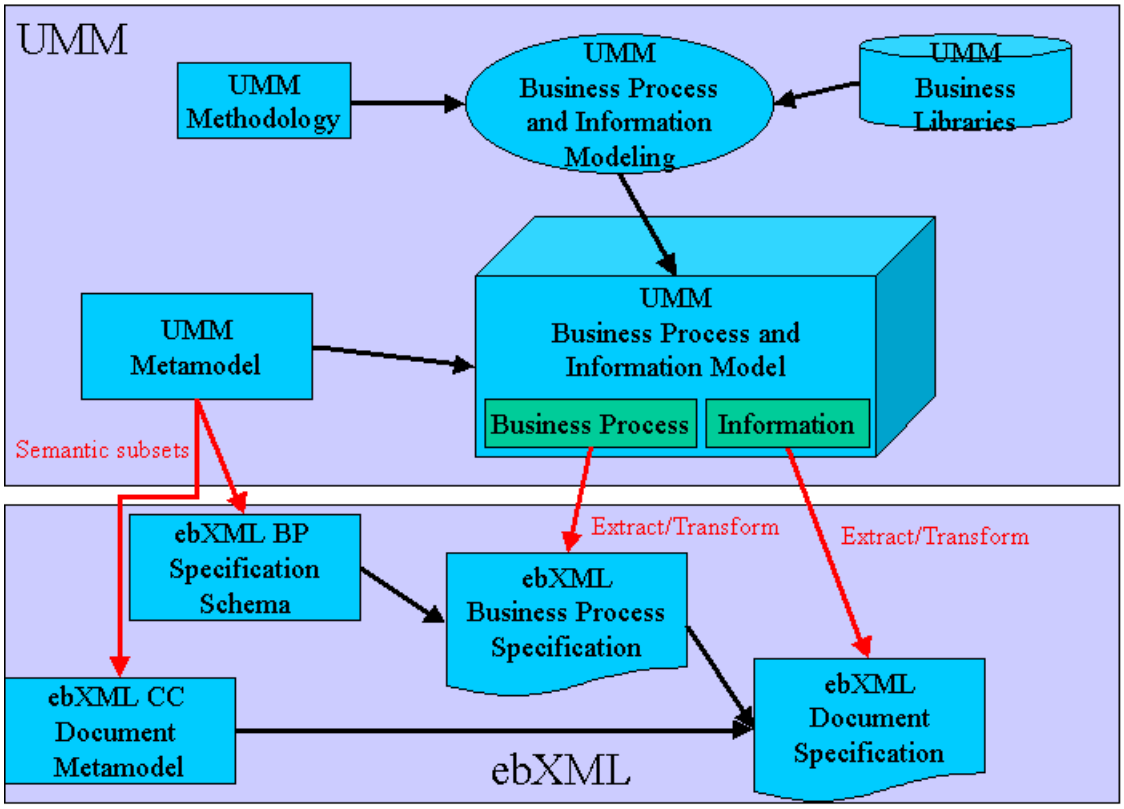
The UMM Meta Model supports a set of Business Process viewpoints that provide a set of semantics (vocabulary) for each viewpoint and forms the basis of specification of the semantics and artifacts that are required to facilitate business process and information integration and interoperability. Using 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.

The *ebXML Business Process Specification Schema* is available in two stand-alone representations, a UML version , and an XML version. The XML version is intended to be interpretable by ebXML compliant software.

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

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



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

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

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

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

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

6 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.

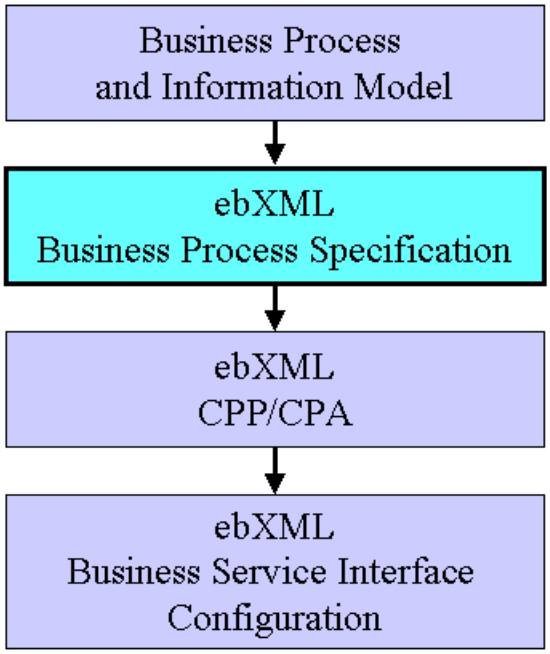


Figure 2: Business Process Specification and Business Service Interface Configuration

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 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.

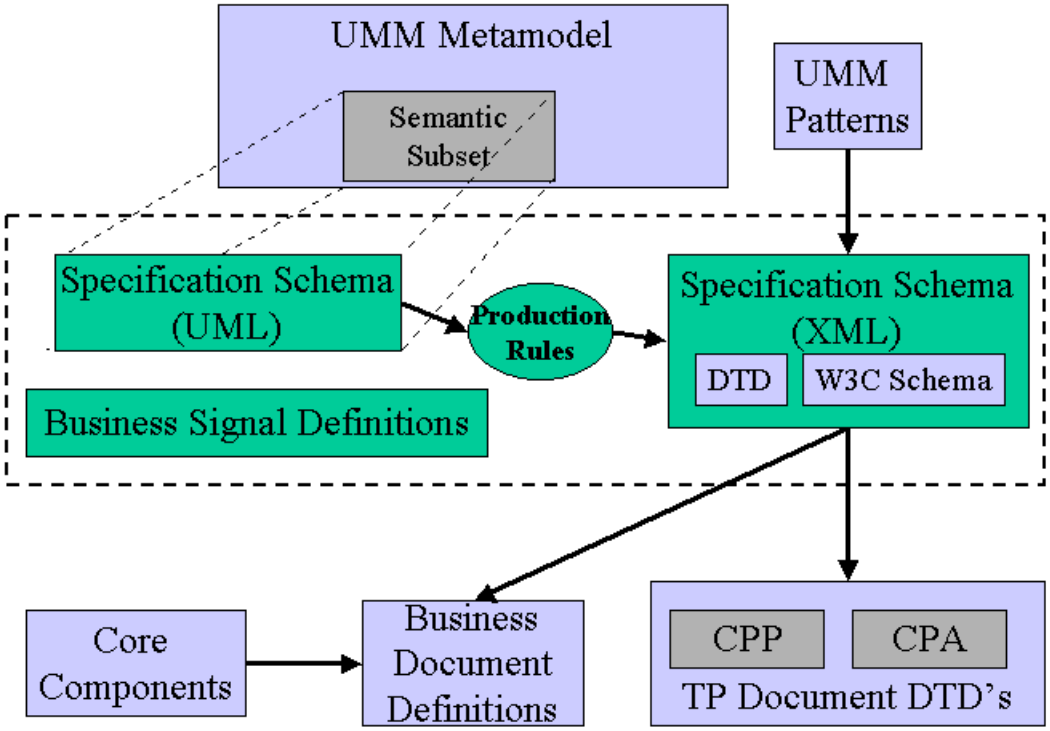


Figure 3: Relationship of *ebXML Business Process Specification Schema* to UMM, CPP/CPA and Core Components

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 version of Business Process Specification Schema

The UML version of the *ebXML Business Process Specification Schema* is a semantic subset of the metamodel behind UMM as specified in UN/CEFACT TMWG's N090R9.1

N090R9.1 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 version of Business Process Specification Schema

The XML version 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 version of the ebXML *Business Process Specification Schema.*

Both a DTD and a W3C Schema is provided. Each is an isomorphic definition of the UML version of the ebXML *Business Process Specification Schema*.

UMM Business Process Interaction Patterns

ebXML Business Service Interfaces are 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 N090R9.1 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.

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 Schema* itself.

The Business Process Specification Schema 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 semantic to the business signals

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

Production Rules

A set of production rules are 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 DTD version of the ebXML *Business Process Specification Schema* from the UML Class Diagram version of the ebXML *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 apply for its conversion into a DTD 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 in essence the machine interpretable run time business process specification needed for an ebXML Business Service Interface. 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* . Within these CPP profiles and CPA agreements are then added 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 definiton 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.

6.1 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.

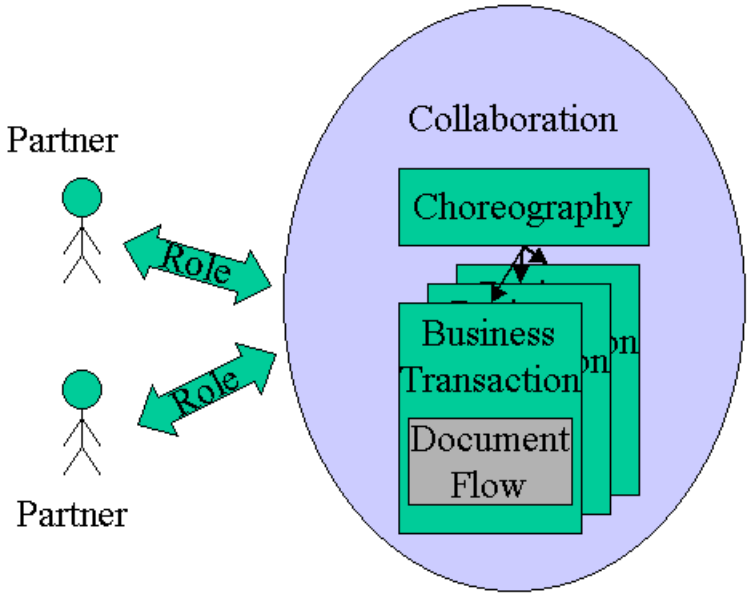


Figure 4. Basic Semantics of a business collaboration

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 a Choreography. Each Business Transaction consists of one or two predefined Business document flows. A Business Transaction may be additionally supported by one or more Business Signals.

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

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.

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 managing the transaction, i.e. an ebXML Business Service Interface (BSI).

A Business Transaction will always either succeed or fail. If it succeeds it may be designated as legally binding between the two partners", or otherwise govern their collaborative activity. If it fails it is null and void, and each partner must relinquish any mutual claim established by the transaction. This can be thought of as ‘rolling back’ the Business Transaction upon failure.

3. Business Document flows

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

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

4. Choreography

   The Business Transaction Choreography describes the ordering and transitions between business transactions or sub collaborations within a binary collaboration. In a UML tool this can be done using a UML activity diagram. The choreography is described in the ebXML *Business Process Specification Schema* using activity diagram concepts such as start state, completion state, activities, synchronizations, transitions between activities, and guards on the transitions.

5. Patterns

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

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

   While the UMM transaction interaction 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 *Business Process Specification Schema*.

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

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

6.2 How to use the ebXML Business Process Specification Schema

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

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

This section discusses

- How the ebXML *Business Process Specification Schema* fits in with other ebXML specifications.
- How to use the ebXML *Business Process Specification Schema* at design time, either for specifying brand new collaborations and transactions, or for re-using existing ones.
- How to specify core transaction semantics and parameters needed for a Collaboration-Protocol Profile and Agreement (CPP/CPA).
- Run-time transaction and collaboration semantics that the ebXML *Business Process Specification Schema* specifies and the Business Service Interface (BSI) is expected to manage.

6.3 How ebXML *Business Process Specification Schema* is used with other ebXML specifications

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

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

As shown in Figure 5, Business Documents are defined at the intersection between the Business Process Specification and the ebXML Core Component specifications. A Business Process Specification will reference, but not define, a set of required Business Documents. At ebXML Business Documents are either defined by some external document specification, or assembled directly or 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.

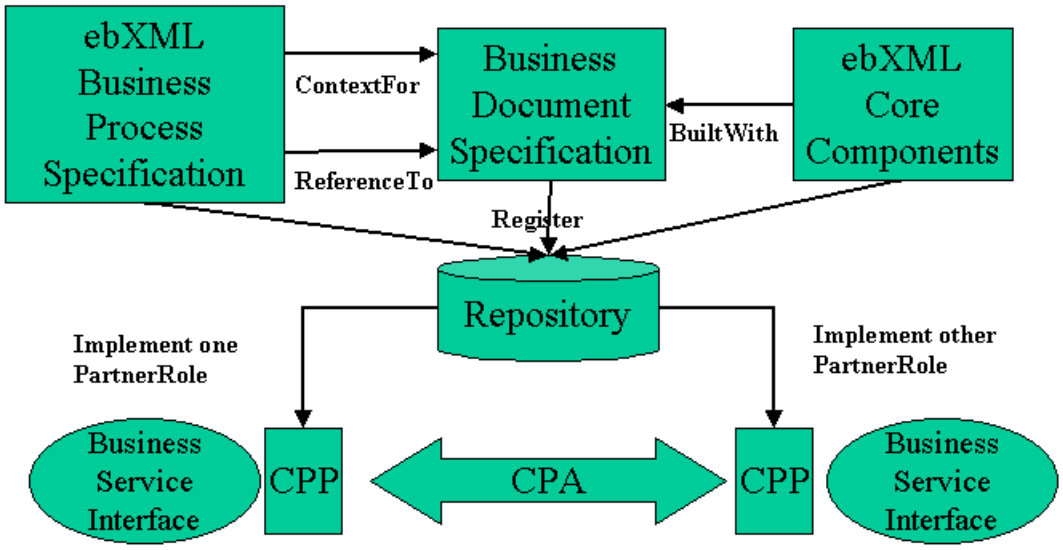


Figure 5: ebXML *Business Process Specification Schema* and other ebXML Specifications

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 version of a *Business Process Specification.*

Alternatively a user would use an XML based tool to produce the XML version 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 version 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.

6.4 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:

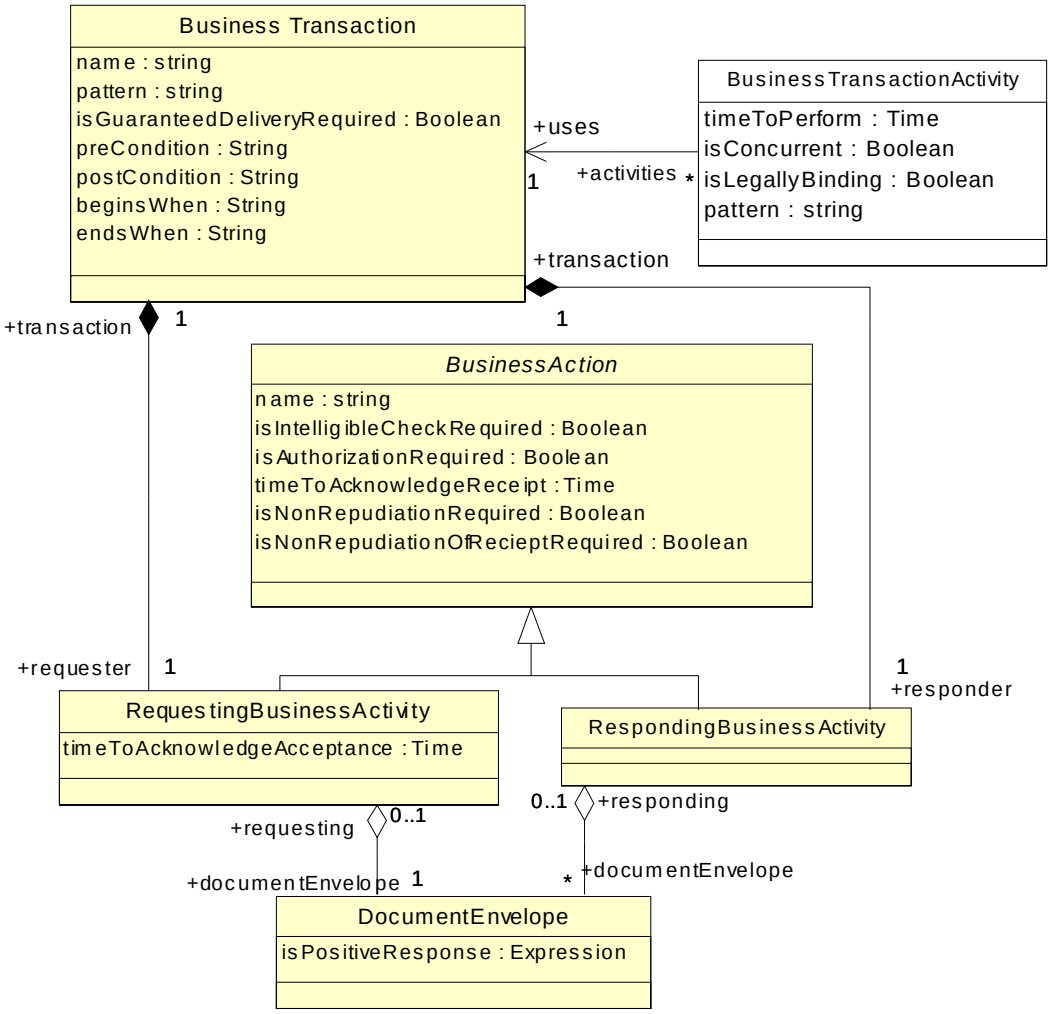
1. Specify a Business Transaction
2. Specify the Business Document flow for a Business Transaction
3. Specify a Binary Collaboration re-using the Business Transaction
4. Specify a Choreography for the Binary Collaboration
5. Specify a higher level Binary Collaboration re-using the lower level Binary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

Although this section, for purposes of introduction, discusses the specification of a 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 Specificaton 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 ebXML Business Process Specification compliance.

6.4.1 Specify a Business Transaction and its Business Document Flow

Figure 6 illustrates a business transaction.



673

674

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

676

677 ### 6.4.1.1 Key Semantics of a Business Transaction

678

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

681 A business transaction consists of a Requesting Business Activity, a
682 Responding Business Activity, and one or two document flows between
683 them. A Business Transaction may be additionally supported by one or
684 more Business Signals that govern the use and meaning of
685 acknowledgements and related matters in the 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 always a Request document flow.

Whether a Response document flow 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.

6.4.1.2 Sample syntax

Here is a simple notification transaction with just one document flow:

```
<BusinessTransaction name="Notify of advanceshipment">
      <RequestingBusinessActivity name="">
            <DocumentEnvelope isPositiveResponse="true"
              BusinessDocument name="ASN"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name=""
      </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.

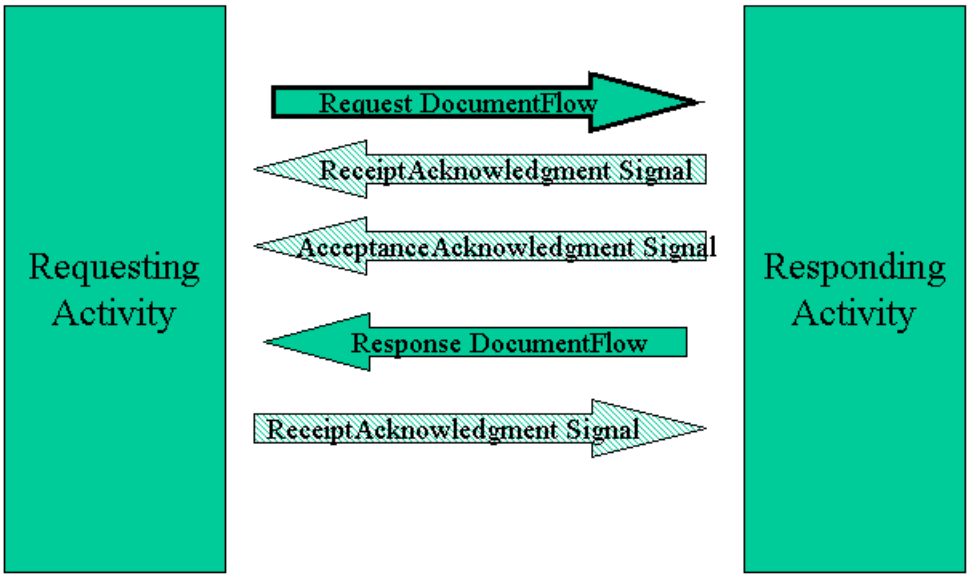


Figure 7: **Possible document flows and signals and their sequence**

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 are 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. The property ***isIntelligibleCheckRequired*** allows partners to agree that a message should be confirmed by a Receipt acknowledgement only if it also is legible. Legible means that it has been passed a 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. This is the case if the contents of the message's business documents and document envelope have passed a business rule validity check.

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 that would have depended on receipt of a business document satisfying the associated timeToPerform.

6.4.1.3 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="Create Order">
      <RequestingBusinessActivity name=""
                  isNonRepudiationRequired="true"
                  timeToAcknowledgeReceipt=”P2D"
                  timeToAcknowledgeAcceptance=”P3D">
            <DocumentEnvelope isPositiveResponse="true"
                  BusinessDocument="Purchase Order"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name=""
                  isNonRepudiationRequired="true"
            timeToAcknowledgeReceipt=”P5D">
            <DocumentEnvelope isPositiveResponse="true"
                  BusinessDocument="PO
            Acknowledgement"/>
            </DocumentEnvelope>
      </RespondingBusinessActivity>
</BusinessTransaction>
```

6.4.1.4 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*.

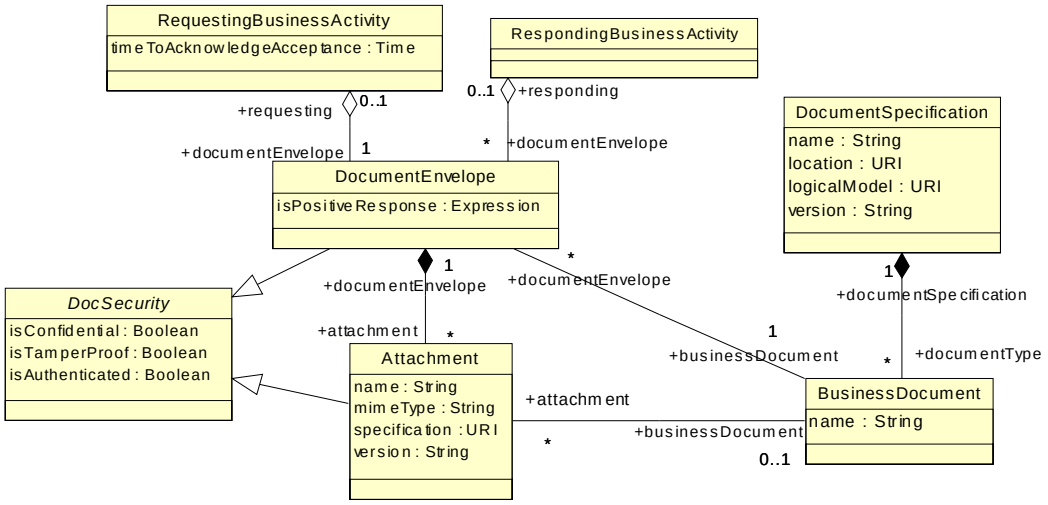


Figure 8: UML Diagram of document flow

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

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

The Document Envelope represents the flow of documents between the activities. Each Document Envelope carries exactly one primary Business Document. A Business Document is defined in a DocumentSpecification. This may be an ebXML DocumentSpecification, or a DocumentSpecification supplied by an outside source.

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.

6.4.1.5 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 documents are defined in a named DocumentSpecification, and the Business Documents are fully qualified with the schema name.

```
<DocumentSpecification name="ebXML1.0" location="someplace"
                    logicalModel="someplaceAlso">
            <BusinessDocument name=" Purchase Order "/>
            <BusinessDocument name=" PO Acknowledgement "/>
            <BusinessDocument name=" PO Rejection "/>
            <BusinessDocument name="Delivery Instructions"/>
        </DocumentSpecification>

        <BusinessTransaction name="Create Order">
            <RequestingBusinessActivity name=""
              <DocumentEnvelope isPositiveResponse="true"
`               BusinessDocument="ebXML1.0/PO Acknowledgement">
                  <Attachment
                        name=”DeliveryNotes”
                        mimeType=”XML”
                        BusinessDocument=
                     "ebXML1.0/Delivery Instructions"
                        specification=””
                        isConfidential=”true”
                        isTamperProof=”true”
                        isAuthenticated=”true”>
                  </Attachment>
              </DocumentEnvelope>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name=""
                    <DocumentEnvelope isPositiveResponse="true"
`                     BusinessDocument="ebXML1.0/PO
Acknowledgement"/>
                    </DocumentEnvelope>
                     <DocumentEnvelope isPositiveResponse="false"
`                     BusinessDocument=" ebXML1.0/PO Rejection"/>
                    </DocumentEnvelope>
            </RespondingBusinessActivity>
        </BusinessTransaction>
```

6.4.2 Specify a Binary Collaboration

Figure 9 illustrates a binary collaboration.

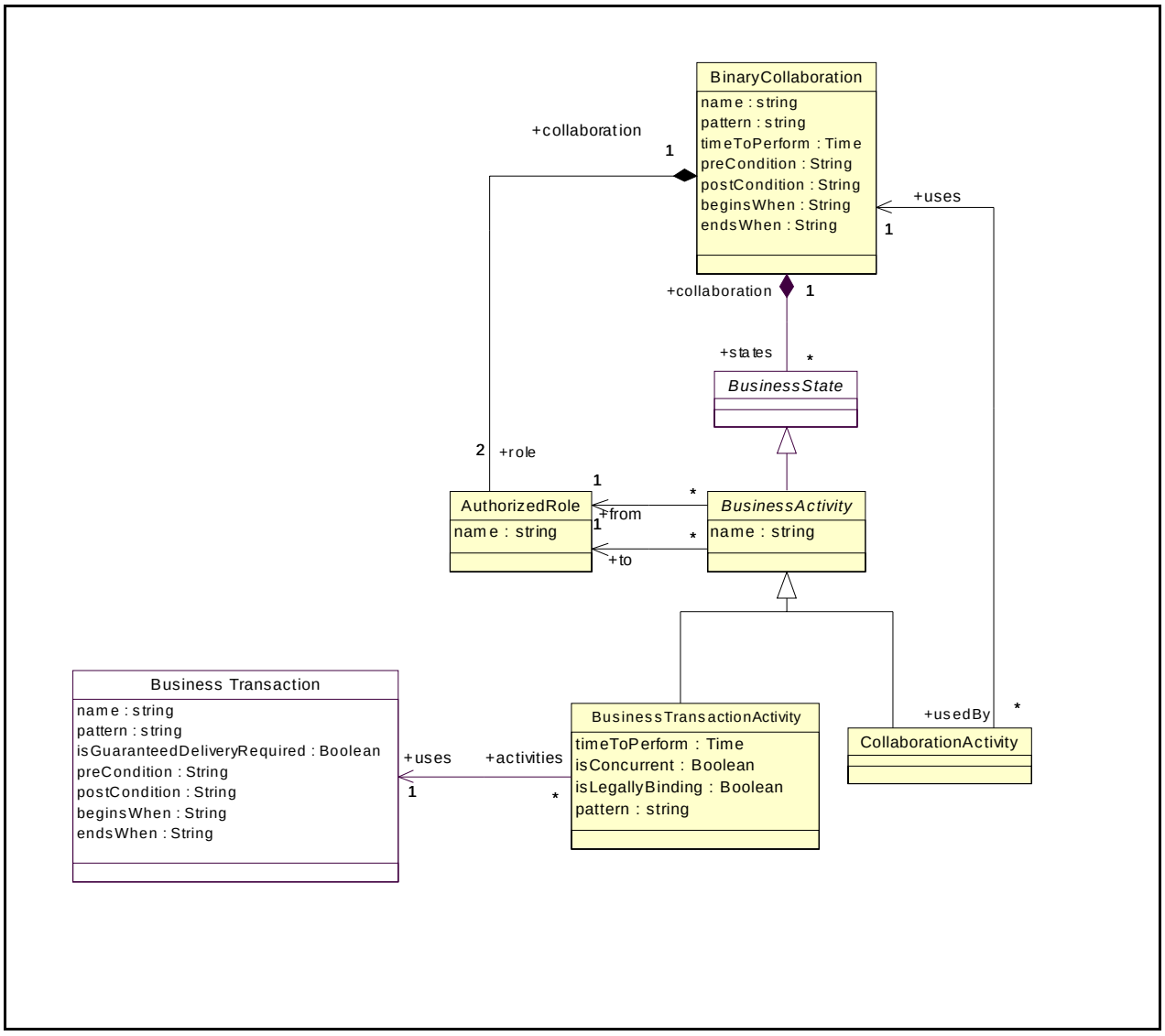


Figure 9: UML Diagram of a Binary Collaboration

6.4.2.1 Key Semantics of a Binary Collaboration

A Binary Collaboration is always between two roles. These two roles are called Authorized Roles, because they represent the actors that are authorized to participate in the collaboration.

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

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.

A Collaboration Activity is the performance of a Binary Collaboration, possibly 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.

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 Authorized roles Customer and Vendor. In Collaboration Activity Q we assign Customer to be the initiator, and Vendor to be the responder. Binary Collaboration X has Authorized 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, 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 requires that parties agree upon a Collaboration Protocol Agreement (CPA) in order to transact business. A CPA associates 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.

6.4.2.2 Sample syntax

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

```
<BinaryCollaboration name="Firm Order”
timeToPerform="P2D">
      <Documentation>
            timeToPerform =
            Period: 2 days from start of transaction
      </Documentation>
      <AuthorizedRole name="buyer"/>
      <AuthorizedRole name="seller"/>
      <BusinessTransactionActivity name="Create Order"
            businessTransaction="Create Order"
      fromAuthorizedRole="buyer"
            toAuthorizedRole="seller"/>
</BinaryCollaboration>
```

Here is a slightly more complex Binary Collaboration re-using the same Business Transaction as the previous Binary Collaboration, and adding the use of another of the Business Transactions defined above.:

```
<BinaryCollaboration name="Product Fulfillment"
timeToPerform="P5D">
      <Documentation>
            timeToPerform =
            Period: 5 days from start of transaction
      </Documentation>
      <AuthorizedRole name="buyer"/>
      <AuthorizedRole name="seller"/>
      <BusinessTransactionActivity name="Create Order"
            businessTransaction="Create Order"
            fromAuthorizedRole="buyer"
            toAuthorizedRole="seller"
            isLegallyBinding=”true” />
      <BusinessTransactionActivity
            name="Notify shipment"
            businessTransaction="Notify of advance
            shipment"
            fromAuthorizedRole="buyer"
            toAuthorizedRole="seller"/>
</BinaryCollaboration>
```

6.4.3 Specify a MultiParty Collaboration

Figure 10 illustrates a multiparty collaboration

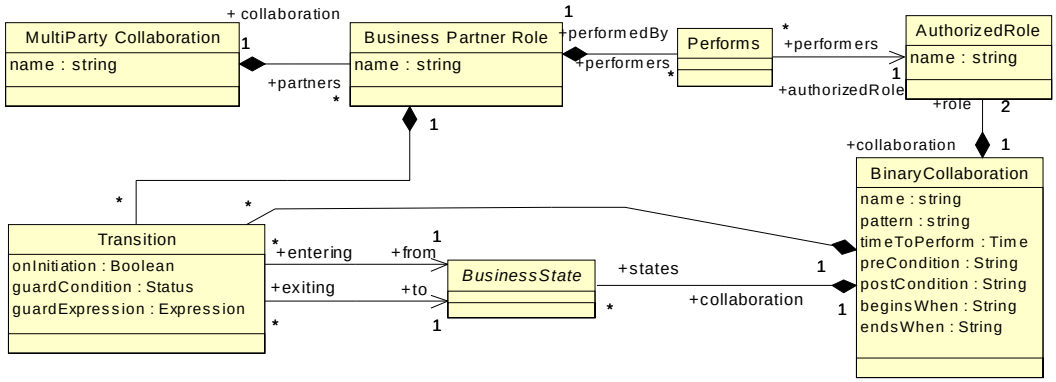


Figure 10: UML Diagram of a MultiParty Collaboration

6.4.3.1 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 Authorized Role in one of the binary collaborations, or perhaps one Authorized 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 an Authorized 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 will 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.

6.4.3.2 Sample syntax

Here is a simple Multiparty Collaboration using the Binary Collaborations defined above.

```
<MultiPartyCollaboration name="DropShip">
<BusinessPartnerRole name="Customer">
     <Performs
     authorizedRole=
```

```
                    '//binaryCollaboration[@name="Firm Order"]
                    /AuthorizedRole[@name="buyer"]'/>
               </BusinessPartnerRole>
               <BusinessPartnerRole name="Retailer">
                    <Performs
                    authorizedRole=
                    '//binaryCollaboration[@name="Firm Order"]
                    /AuthorizedRole[@name="seller"]'/>
                    <Performs
                    '//binaryCollaboration[@name=" Product
                    Fulfillment" /AuthorizedRole[@name="buyer"]'/>
               </BusinessPartnerRole>
               <BusinessPartnerRole name="DropShip Vendor">
                    <Performs
                      '//binaryCollaboration[@name=" Product
                    Fulfillment"
                    /AuthorizedRole[@name="seller"]'/>
               </BusinessPartnerRole>
          </MultiPartyCollaboration>
```

6.4.4 Specify a Choreography

Figure 11 illustrates a choreography.

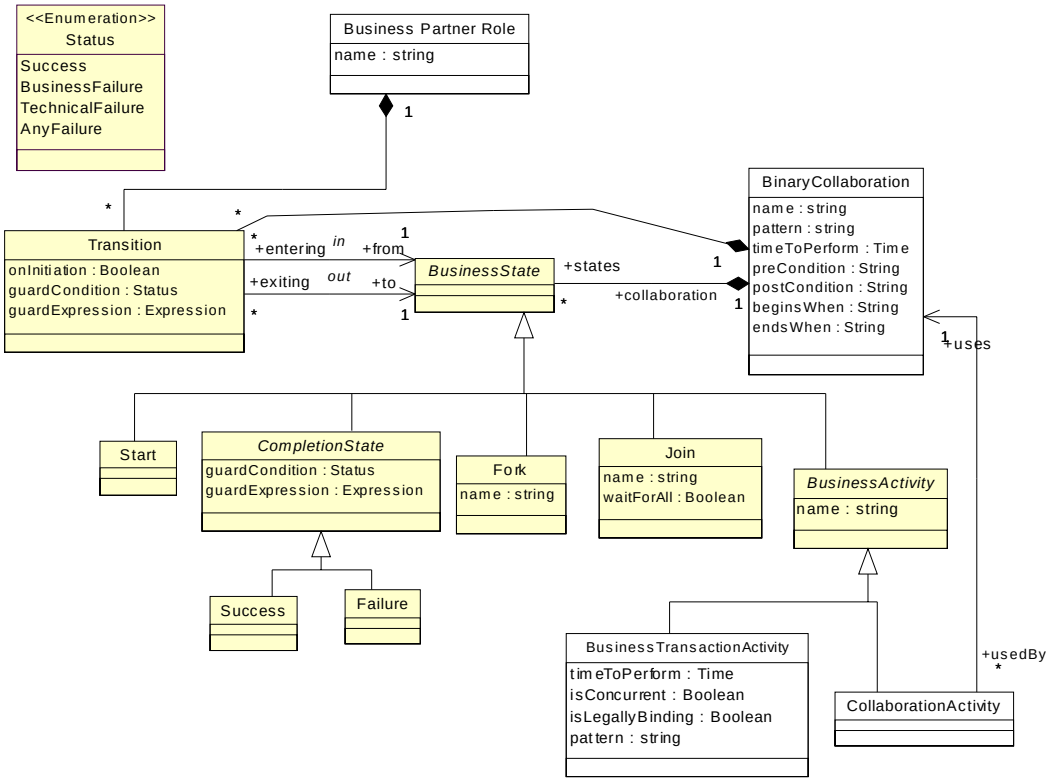


Figure 11: UML Diagram of a Choreography

6.4.4.1 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.

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 Synchronization state. These are all equivalent to diagramming artifacts on a UML activity chart.

Transitions are between Business States. Transitions can be gated by Guards. Guards can refer to the status of the Document Envelope that caused the transition, the type of Document sent, the content of the document, or postconditions on the prior state.

A Transition can also be used to create nested BusinessTransactionActivities. 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 an Authorized Role in one Binary Collaboration receives a request, then turns around and becomes the requestor in an 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.

6.4.4.2 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="Product Fulfillment"
timeToPerform="P5D">
      <Documentation>
            timeToPerform =
            Period: 5 days from start of transaction
      </Documentation>
      <AuthorizedRole name="buyer"/>
      <AuthorizedRole name="seller"/>
      <BusinessTransactionActivity name="Create Order"
            businessTransaction="Create Order"
            fromAuthorizedRole="buyer"
            toAuthorizedRole="seller"/>
      <BusinessTransactionActivity
            name="Notify shipment"
            businessTransaction="Notify of advance
            shipment"
            fromAuthorizedRole="buyer"
```

```
                    toAuthorizedRole="seller"/>
               <Start toBusinessState="Create Order"/>
               <Transition
                    fromBusinessState="Create Order"
                    toBusinessState="Notify shipment"/>
               <Success fromBusinessState="Notify shipment"
                    guardCondition="Success"/>
               <Failure fromBusinessState="Notify shipment"
                    guardCondition="BusinessFailure"/>
          </BinaryCollaboration>
```

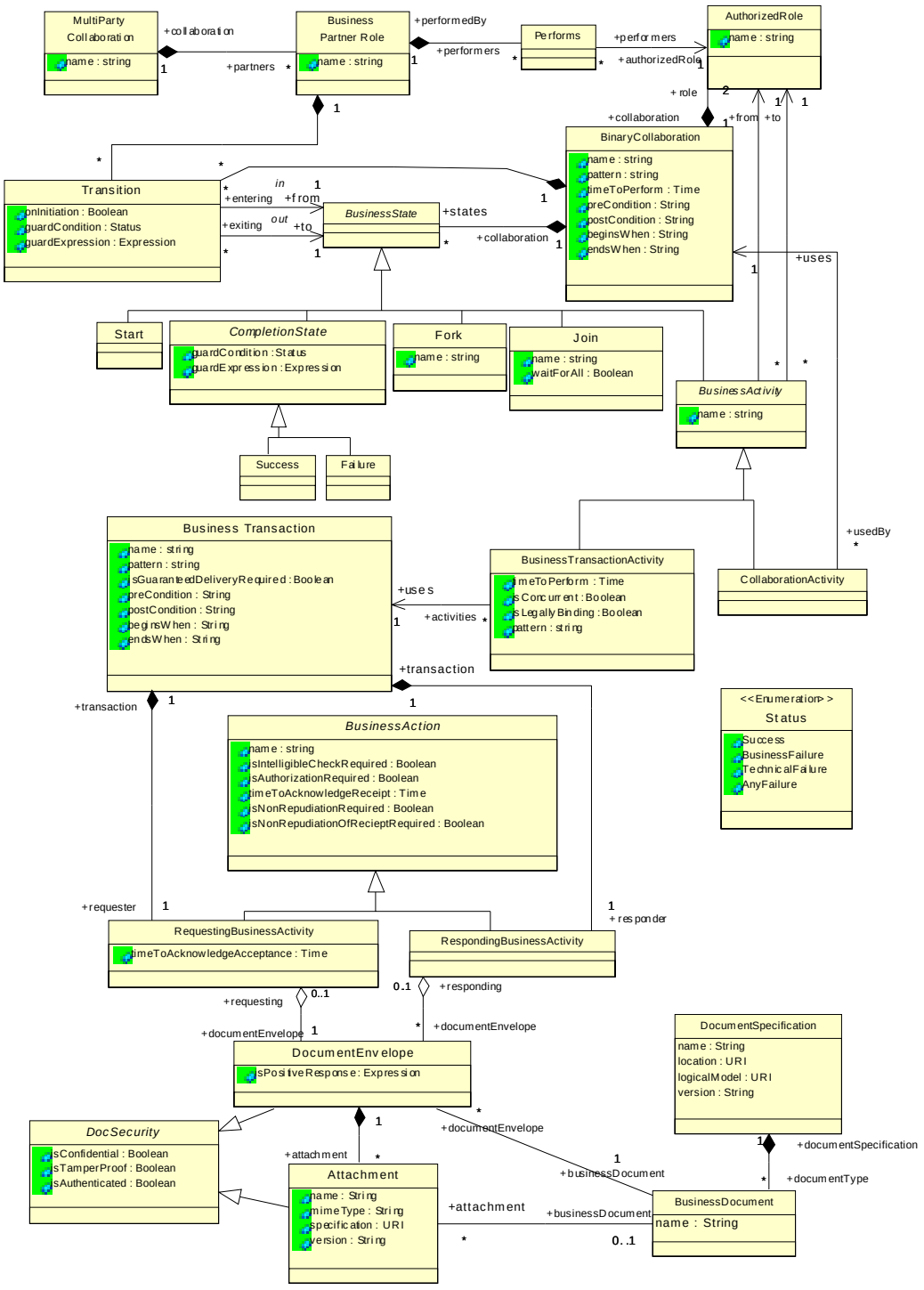
Here is the same Multiparty Collaboration as defined before, but with a simple choreography (transition) across two Binary Collaborations.

```
<MultiPartyCollaboration name="DropShip">
          <BusinessPartnerRole name="Customer">
               <Performs
               authorizedRole=
               ‘//binaryCollaboration[@name="Firm Order”]
               /AuthorizedRole[@name=”buyer”]’/>
          </BusinessPartnerRole>
          <BusinessPartnerRole name="Retailer">
               <Performs
               authorizedRole=
               ‘//binaryCollaboration[@name="Firm Order”]
               /AuthorizedRole[@name=”seller”]’/>
               <Performs
               ‘//binaryCollaboration[@name=" Product
               Fulfillment” /AuthorizedRole[@name=”buyer”]’/>
               <Transition
               fromBinaryCollaboration”Firm Order”
               fromBusinessState=
               ‘//binaryCollaboration[@name="Firm Order”]
               /[@name="Create Order"]’
               toBusinessState=
               ‘//binaryCollaboration[@name="Product
               Fulfillment”]
               /[@name="Create Order"]’
               />
          </BusinessPartnerRole>
          <BusinessPartnerRole name="DropShip Vendor">
               <Performs
                 ‘//binaryCollaboration[@name=" Product
               Fulfillment”
               /AuthorizedRole[@name=”seller”]’/>
          </BusinessPartnerRole>
     </MultiPartyCollaboration>
```

6.4.5 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*

Figure 12: Overall ebXML *Business Process Specification Schema* as UML class diagram



6.5 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.
- 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 athorization 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

6.5.1 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 N090R9.1 chapter on the UMM metamodel has a detail state chart for each of the business partners.

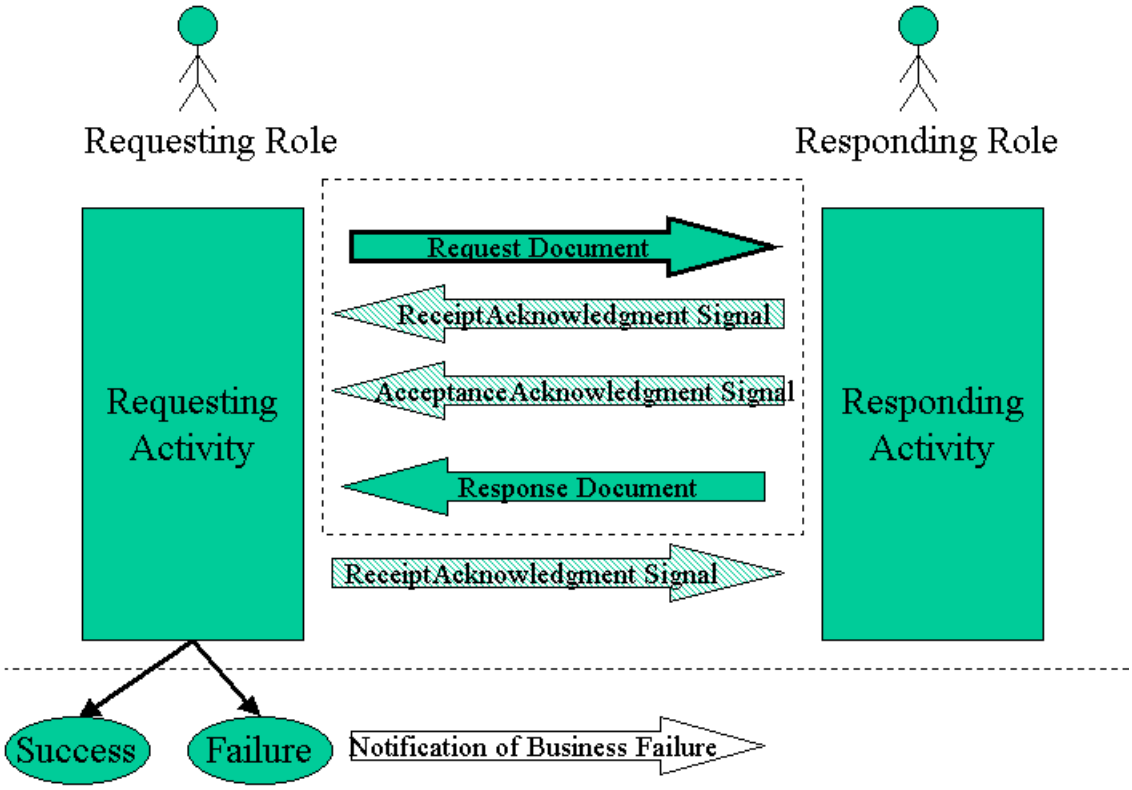


Figure 13: Schematic of core Business Transaction semantics.

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 the transaction must be rolled back to a defined state before the transaction was initiated.
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 (ir 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 receipt or non-receipt of the request, the response and/or business signals
    b. The occurrence of time-outs
    c. The occurrence of a business exception
    d. The occurrence of a control exception
    e. The interpretation of the received response and guard expressions on transitions to success or failure
10. The determination of Business Transaction success or failure is established by the requesting party based on the above success or failure factors. 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 signal back to the responding role. This is merely a signal and does not pass control back to the responding activity, nor does it alter the successful or failed completion of the Business Transaction that was based on the receipt of the Response.
12. Upon identifying a time-out exception in the processing of a Business Transaction, and closing the transaction accordingly, the requesting party may send a notification of failure to the responding party. This is considered a new Business Transaction and does not alter the already established conclusion of the Business Transaction.

6.5.1.1 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 is 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.

6.5.2 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 this 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 and Simple Negotiation Patterns.

6.5.3 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 than 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, 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 affect this schema's determination of a "success" or "failure" end state..

The other requires the responder to send a signed copy of the receipt, which the requestor then saves. This is the isNonRepudiationOfReceiptRequired parameter in the requesting 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 affect this schema's 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.

Parameter	BSI requirement
isNonRepudiationRequired	Must save audit trail of messages it sends
isNonRepudiationOfReceiptRequired	Must digitally sign receiptAcknowledgements

6.5.4 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.

Parameter	BSI requirement
IsAuthorizationRequired	Must validate identity of originator against a list of authorized originators

IsAuthorizationRequired is specified on the requesting and responding activity accordingly.

6.5.5 Document security

The following security characteristics of each Business Document being transported, even if many are collected in the same message, can be specified individually, or collectively within a Document Envelope:

Parameter	Delivery Channel requirement
isConfidential.	The information entity is encrypted so that unauthorized parties cannot view the information
isTamperProof.	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.
isAuthenticated.	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.

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.

When set to YES (or TRUE) these parameters assume that the corresponding security characteristic is provided in a manner providing persistence. Compliance requires that the specified character of the document survive its reception at a business service interface, and persist as the document is archived or forwarded.

6.5.6 Reliability

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

Parameter	Delivery Channel requirement
IsGuaranteedDeliveryRequired	This means that Business Documents transferred are guaranteed (by some delivery channel or other party other than the trading partners) to be delivered

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

6.5.7 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.

6.6 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 guardCondition and guardExpression for determination of success
4. Detection of failed completion of a transaction
    a. Detection of time-outs
    b. Detection of exceptions
    c. Application of business rules expressed as isPositiveResponse and transition guardCondition and guardExpression for determination of failure
5. Notification of failure
6. Receipt of notification of failure
7. Rollback upon failure (note this is the independent responsibility of each role, it is not a co-coordinated roll-back, there are no 2-phase commits in ebXML)

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 do the necessary roll-back, and to exit the transaction. The responsibilities of the two roles differ slightly and are described in each of the sections below. Generally, if a failure happens at the responding role, the responding role will send an exception signal to the requesting role, and both parties will exit the current transaction. If a failure happens at the requesting role, the requesting role will exit the current transaction and in a separate transaction notify the responding role about the failure. This way the flow of control within a transaction is always unambiguous and finite.

6.6.1 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 role has to acknowledge receipt of a business document.
Acceptance Acknowledgement (Non-substantive)	timeToAcknowledgeAcceptance	The time a responding 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.

A requesting partner terminates if a timeout is thrown and then sends a notification of failure to the responder as part of a separate transaction.

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.

6.6.2 Exceptions

Under all normal circumstances the response message and/or the time-outs determine the success or failure of a business transaction. However the business processing of the transaction can go wrong at either the responding or the requesting role.

6.6.2.1 ControlException

A *ControlException* signals an error condition in the management of a business transaction. This business signal is asynchronously 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 structure, syntax and message element values.

6.6.2.2 Business Protocol Exceptions

A Business Protocol Exception (or *ProcessException)* signals an error condition in a business activity. This business signal is asynchronously 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 and validation. 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. This type of exception is usually generated after an *AcceptanceAcknowledgement* has been returned.

A business protocol exception terminates the business transaction. The following are business protocol exceptions.

- Negative acknowledgement of receipt. The structure/schema of a message is invalid.
- Negative acknowledgement of acceptance. The business rules are violated.
- Performance exceptions. The requested business action cannot be performed.
- Sequence exceptions. The order or type of a business document or business signal is incorrect.
- 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.
- Business process control exceptions. Business documents are not signed for non-repudiation when required.

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. To preserve this semantics, control failures and business failures are treated differently by the requesting and responding roles as follows:

A responding role that encounters a business protocol exception signals the exception back to the requesting role and then terminates the business transaction. If any business exceptions (includes negative receipt and acceptance acknowledgements) are signaled then the business transaction must terminate.

A requesting role that encounters a business protocol exception terminates the transaction but does NOT send a business exception signal to the responding role. Rather, the requesting role then sends as a separate Business Transaction a notification revoking the offending business document request. This new transaction may be defined as a continuation of the current Binary Collaboration, or it may start a new Binary Collaboration specifically defined to handle this notification of failure.

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 if the sending partner role is not authorized to perform the business activity. A sending partner must send notification of failed authorization if a responding 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 business control exception if the sending partner role has not properly delivered their business document. A requesting partner must send notification of failed business control 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-repudiatable. A requesting partner must send notification of failed business control (possibly revoking a contractual offer) if a responding partner has not properly delivered their business document. 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. 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.

If a requesting role, upon evaluation of these expressions, determines a failure, then the requesting role will “roll back” the Business Transaction and send a notification of failure.

6.7 Runtime Collaboration Semantics

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

6.8 Where the ebXML Business Process Specification Schema May Be Implemented

The ebXML *Business Process Specification Schema* should be used wherever software is being specified to perform a role in an ebXML business collaboration. Specifically, the ebXML *Business Process Specification Schema* is intended to provide the business process and document specification for the formation of 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.

7 UML Element Specification

In the following we will review all the specification elements in the UML version of the ebXML *Business Process Specification Schema*, grouped as follows:

- Business Collaborations
  - Multiparty
  - Binary
- Business Transactions
- Document flow
- Choreography

7.1 Business Collaborations

7.1.1 MultiPartyCollaboration

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.

Tagged Values:

name.	Defines the name of the MultiPartyCollaboration

Associations:

partners	A multiparty collaboration has two or more BusinessPartnerRoles

Wellformedness Rules:

All multiparty collaborations must be synthesized from binary collaborations

7.1.2 BusinessPartnerRole

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

Tagged Values:

name.	Defines the name of the role played by partner in the overall multiparty business collaboration, e.g. customer or supplier.

Associations:

performers.	The Authorized Roles performed by a partner in the binary business collaboration.
transitions	The transitions (managed by this BusinessPartnerRole) between activities across binary collaborations
collaboration	The Business Partner Role participates in one multi party collaboration

Wellformedness Rules:

A partner must not perform both roles in a given business activity.

7.1.3 Performs

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.

Tagged Values:

NONE

Associations:

performedBy	An instance of Performs is performed by only one BusinessPartnerRole
authorizedRole	The AuthorizedRole that will be performed by the Business PartnerRole

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.

7.1.4 AuthorizedRole

An Authorized Role is a role that is authorized to send the request or response, e.g. the buyer is authorized to send the request for purchase order, the seller is authorized to send the acceptance of purchase order.

Tagged Values:

name	Defines the name of the AuthorizedRole uniquely within the Binary Collaboration

Associations:

performers	An AuthorizedRole may be used by one or more performers, i.e. Business Partner Roles in a multiparty collaboration
from	An AuthorizedRole may be the initiator in a business activity
to	An AuthorizedRole may be the responder in a business activity
collaboration	An AuthorizedRole may be in only one BinaryCollaboration

Wellformedness Rules:

An AuthorizedRole may not be both the requestor and the responder in a business transaction

An AuthorizedRole may not be both the initiator and the responder in a binary business collaboration

7.1.5 BinaryCollaboration

A Binary Collaboration defines a protocol of interaction between two authorized roles.

A 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.

A Binary Collaboration choreographs one or more business transaction activities between two roles.

A Binary Collaboration is not an atomic transaction and should not be used in cases where Business Transaction rollback is required.

Tagged Values:

name	Defines the name of the BinaryCollaboration
timeToPerform	The period of time, starting upon initiation of the first activity, within which this entire collaboration must conclude.
preCondition	A description of a state external to this collaboration that is required before this collaboration can commence.
postCondition	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.
beginsWhen	A description of an event external to the collaboration that normally causes this collaboration to commence.
endsWhen	A description of an event external to this collaboration that normally causes this collaboration to conclude.
pattern	The optional reference to a pattern that this binary collaboration is based on

Associations:

role	A binary collaboration consists of two authorized roles
states	A binary collaboration consists of one or more states, some of which are ‘static’, and some of which are action states
usedBy	A binary collaboration may be used within another binary collaboration via a collaboration activity
transitions	The transitions between activities in this binary collaboration

Wellformedness Rules:

NONE

7.1.6 BusinessActivity

A business activity is an action state within a binary collaboration. It is the super type for BusinessTransactionActivity and CollaborationActivity, specifying the activity of performing a transaction or another binary collaboration respectively.

Supertype of:

BusinessActionActivity, CollaborationActivity

Subtype of:

BusinessState

Tagged Values:

name	Defines the name of the activity uniquely within the binary collaboration

Associations:

from	The initiating role
to	The responding role

Wellformedness Rules:

NONE

7.1.7 BusinessTransactionActivity

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***.

Subtype of:

BusinessActivity

Tagged Values:

timeToPerform	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.
isConcurrent.	If the BusinessTransactionActivity is concurrent then more than one instance of the associated BusinessTransaction can be performed as part

of the execution of this BusinesTransactionActivity

isLegallyBinding Defines whether the Business Transaction performed by this activity is intended by the trading parties to be binding. Default value is True.

Associations:

uses. The business transaction activity performs (uses) exactly one business transaction.

Wellformedness Rules:

NONE

7.1.8 CollaborationActivity

A collaboration activity is the activity of performing a binary collaboration within another binary collaboration.

Subtype of:

BusinessActivity

Tagged Values:

NONE (other than inherited)

Associations:

uses A collaboration activity uses exactly one binary collaboration

Wellformedness Rules:

A binary collaboration may not re-use itself

7.2 Business Transactions

7.2.1 BusinessTransaction

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.

Tagged Values:

name	Defines the name of the Business Transaction.
isGuaranteedDeliveryRequired.	Both partners must agree to use a transport that guarantees delivery
preCondition	A description of a state external to this transaction that is required before this transaction can commence.
postCondition	A description of a state that does not exist before the execution of this transaction but will exist as a result of the execution of this transaction.
beginsWhen	A description of an event external to the transaction that normally causes this transaction to commence.
endsWhen	A description of an event external to this transaction that normally causes this transaction to conclude.
pattern	The optional reference to a pattern that this transaction is based on.

Associations:

activities	A BusinessTransaction can be performed by many BusinessTransactionActivites
requester	A BusinessTransaction has exactly one RequestingBusinessActivity
responder	A BusinessTransaction has exactly one RespondingBusinessActivity

Wellformedness Rules:

NONE

7.2.2 Business Action

A Business Action is an abstract super class. Business Action, is the holder of attributes that are common to both Requesting Business Activity and Responding Business Activity.

Tagged Values:

name	Defines the name of the RequestingBusinessTransaction or

RespondingBusinessTransaction depending on the subtype

IsAuthorizationRequired 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)

IsNonRepudiationRequired 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)

isNonRepudiationOfReceiptRequired. 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)

timeToAcknowledgeReceipt The time a receiving role has to acknowledge receipt of a business document. This parameter is specified on the sending side. (See also section on core transaction semantics)

isIntelligibleCheckRequired 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)

Associations:

NONE

Wellformedness Rules:

NONE

7.2.3 RequestingBusinessActivity

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.

Subtype of:

BusinessAction

Tagged Values:

timeToAcknowledgeAcceptance 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)

Associations:

transaction A requesting activity is performed in exactly one business transaction

documentEnvelope A requesting activity sends exactly one Document Envelope

Wellformedness Rules:

NONE

7.2.4 RespondingBusinessActivity

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.

Subtype of:

BusinessAction

Tagged Values:

NONE, except as inherited from Business Action

Associations:

transaction A responding activity is performed in exactly one business transaction

DocumentEnvelope A responding activity may specify zero or more but sends at most one Document Envelope

Wellformedness Rules:

NONE

7.3 Document flow

7.3.1 Document Security

DocumentSecurity is an abstract super class holding the security related attributes for DocumentEnvelope and Attachment.

Supertype of:

DocumentEnvelope and Attachment

Tagged Values:

IsAuthenticated	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	The information entity is encrypted so that unauthorized parties cannot view the information. (See also section on Document Security)
isTamperProof	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)

Associations:

NONE

Wellformedness Rules:

NONE

7.3.2 Document Envelope

A Document Envelope is what conveys business information between the two roles in a business transaction. One Document Envelope conveys the request from the requesting role to the responding role, and another Document Envelope conveys the response (if any) from the responding role back to the requesting role.

Subtype of:

DocumentSecurity

Tagged Values:

isPositiveResponse An expression whose evaluation results in TRUE or FALSE. If TRUE this DocumentEnvelope is intended as a positive response to the request. This parameter is only relevant on the response envelope. Its value does not bind the recipient, or override the computation of transactional success or failure using the transaction's guard expressions.

Associations:

requesting This is a reference to the requesting activity associated with this DocumentEnvelope. This requesting activity may be the sender, or the receiver depending on whether the DocumentEnvelope represents a request or a response.

responding This is a reference to the requesting activity associated with this DocumentEnvelope. This responding activity may be the sender, or the receiver depending on whether the DocumentEnvelope represents a request or a response.

BusinessDocument This identifies the primary Business Document in the envelope. A Document Envelope contains exactly one primary Business Document.

attachment A Document Envelope contains an optional set of attachments related to the 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 .

.

7.3.3 BusinessDocument

BusinessDocument is a generic name of a document. The location of the definition of the document can be found in the associated DocumentSpecification.

Tagged Values:

name Defines the generic name of the Business Document as it is known within this Business Process Specification

Associations:

documentSpecification A Business Document is in at most one DocumentSpecification

documentEnvelope A Business Document can be in multiple Document Envelopes

attachment A Business Document can serve to specify the type of many attachments

Wellformedness Rules:

NONE

7.3.4 DocumentSpecification

A DocumentSpecification is a collection of Document Definitions. The DocumentSpecification is usually external to the process specification, and is referenced with a URI. An additional reference is to where the logical model is for the Documents in the DocumentSpecification. Typically this would be an ebXML core component context model.

Tagged Values:

name Defines the generic name of the DocumentSpecification as it is known within this Business Process Specification

location Reference to an external source of the DocumentSpecification definition

logicalModel Reference is to where the logical model is

version The version of the DocumentSpecification

Associations:

businessDocument A DocumentSpecification defines many document types

Wellformedness Rules:

NONE

7.3.5 Attachment

Attachment is an optional attachment to a BusinessDocument in a Document Envelope

Subtype of:

DocumentSecurity

Tagged Values:

name	Defines the name of the attachment
mimeType	Defines the valid MIME (Multipurpose Internet Mail Extensions) type of this Attachment
specification	A reference to an external source of description of this attachment.
version	The version of the Attachment

Associations:

documentEnvelope	An Attachment is in exactly one Document Envelope
businessDocument	An Attachment can be defined by a BusinessDocument. If it is not of a defined Business Document, the mime type and spec will be the only indication of its type.

Wellformedness Rules:

NONE

7.4 Choreography within Collaborations.

7.4.1 BusinessState

A business state is any state that a binary collaboration can be in. Start and CompletionState are a snapshot right before or right after an activity, BusinessActivity is an action states that denote the state of being in an activity. Fork and Join reflect the activity of forking to multiple activities or joining back from them.

Supertype of:

Start, CompletionState, Fork, Join, BusinessActivity

Tagged Values:

none

Associations:

collaboration	A business state belongs to only one binary collaboration

entering	A transition that reflects entry into this state
exiting	*A* transition that reflects exiting from this state

Wellformedness Rules:

NONE

7.4.2 Transition

A transition is a transition between two business states in a binary collaboration.

Choreography is expressed as transitions between business states

Tagged Values:

onInitiation	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.
guardCondition	A reference to the status of the previous transaction. A fixed value of Success, BusinessFailure, TechnicalFailure, or AnyFailure
guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition should happen or not. The expression can refer to the name or content of the the most recent DocumentEnvelope or content of documents within it.

Associations:

in	The business state this transition is entering
out	The business state this transition is exiting

Wellformedness Rules:

A transition cannot enter and exit the same state

7.4.3 Start

The starting state for an Binary Collaboration. A Binary Collaboration should have at least one starting activity. If none defined, then all activities are considered allowable entry points.

Subtype of:

BusinessState

Tagged Values:

NONE

Associations:

NONE

Wellformedness Rules:

NONE

7.4.4 CompletionState

The ending state of an binary collaboration, sub classed by success and failure

Supertype of:

Success, Failure

Subtype of:

BusinessState

CompletionState

Tagged Values:

NONE

Associations:

NONE

Wellformedness Rules:

NONE

7.4.5 Success

Defines the successful conclusion of a binary collaboration as a transition from an activity.

Subtype of:

CompletionState

Tagged Values:

NONE, except as inherited

Associations:

NONE

Wellformedness Rules:

Every activity Binary Collaboration should have at least one success

7.4.6 Failure

A subtype of CompletionState which defines the unsuccessful conclusion of a binary collaboration as a transition from an activity.

Subtype of:

CompletionState

Tagged Values:

NONE, except as inherited

Associations:

None

Wellformedness Rules:

Every Binary Collaboration should have at least one failure

7.4.7 Fork

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.

Subtype of:

BusinessState

Tagged Values:

Name	Defines the name of the Fork state

Associations:

None

Wellformedness Rules:

None

7.4.8 Join

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.

Subtype of:

BusinessState

Tagged Values:

Name	Defines the name of the Join state
waitForAll	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.

Associations:

None

Wellformedness Rules:

None

7.5 Definition and Scope

The ebXML *Business Process Specification Schema* should be used wherever software is being specified to perform a role in an ebXML binary collaboration. Specifically, the ebXML *Business Process Specification Schema* is intended to provide the business process and document specification for the formation of a trading partner Collaboration Protocol Profile and Agreement. A set of specification rules have been established to properly constrain the expression of a business process and information model in a way that can be directly incorporated into a trading partner Collaboration Protocol Profile and Agreement.

7.6 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.

[7] If non-repudiation is required at the requesting business activity, then there must be a responding business document.

[8] The time to acknowledge receipt, time to acknowledge acceptance and time to perform properties must be specified for both the requesting and responding business activities and they must be equal.

RequestingBusinessActivity

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

[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 successfully performed.

[11] 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.

[12] 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.

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

[14] 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

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

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

Business Collaboration

[17] A Business Partner Role cannot provide both the initiating and responding roles of the same business transaction activity.

8 ebXML Business Process Specification Schema – (DTD)

In this section we describe the DTD and XML Schema version of the Specification Schema. There are minimal differences between the DTD and the XML Schema, therefore the elements will only be described once, noting differences when needed. This discussion includes

- An example XML Business Process Specification listed in Appendix A.
- A listing of the DTD in Appendix B and the XML Schema in Appendix C
- A table listing all the elements with definitions and parent/child relationships
- A table listing all the attributes with definitions and parent element 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 DTD

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

Overall Structure excluding attribute definitions:

```
ProcessSpecification (Documentation*, (Include* | DocumentSpecification* |
                      ProcessSpecification* | Package | BinaryCollaboration |
                      BusinessTransaction | MultiPartyCollaboration)*)
  Documentation()
  Include( Documentation* )
  DocumentSpecification( Documentation*, BusinessDocument* )
    BusinessDocument( Documentation* )
      Package( Documentation*, (Package | BinaryCollaboration |
               BusinessTransaction | MultiPartyCollaboration)* )
    BinaryCollaboration( Documentation*, AuthorizedRole, AuthorizedRole,
                         (Documentation* | Start | Transition | Success | Failure |
                          BusinessTransactionActivity | CollaborationActivity | Fork | Join)*)
      AuthorizedRole( Documentation* )
      Start( Documentation* )
      Transition( Documentation* )
      Success( Documentation* )
      Failure( Documentation* )
      Fork( Documentation* )
      Join( Documentation* )
      BusinessTransactionActivity( Documentation* )
      CollaborationActivity( Documentation* )
    BusinessTransaction( Documentation*, RequestingBusinessActivity,
```

RespondingBusinessActivity)
RequestingBusinessActivity(Documentation*, DocumentEnvelope)
RespondingBusinessActivity(Documentation*, DocumentEnvelope*)
MultiPartyCollaboration(Documentation*, BusinessPartnerRole*)
BusinessPartnerRole(Documentation*, Performs*, Transition*)
Performs(Documentation*)
Transition(Documentation*)

a. Attachment

XML Element Name: Attachment

DTD Declaration:

```
<!ELEMENT Attachment (Documentation*)>
<!ATTLIST Attachment
      name                 CDATA #REQUIRED
      nameID               ID    #IMPLIED
      BusinessDocument     CDATA #IMPLIED
      BusinessDocumentIDRef IDREF #IMPLIED
      mimeType             CDATA #IMPLIED
      specification        CDATA #IMPLIED
      version              CDATA #IMPLIED
      isConfidential       (true | false) "false"
      isTamperProof        (true | false) "false"
      isAuthenticated      (true | false) "false">
```

Definition:

An optional attachment to a BusinessDocument in a DocumentEnvelope.

Parent Elements:

- DocumentEnvelope

2344 **Attributes:**

Attribute Name	Definition	Default Value
name	Defines the name of the attachment.	Required input
nameID	XML ID version of name	Optional input
businessDocument	An Attachment can be defined by a BusinessDocument. If it is not of a defined Business Document, the mime type and spec will be the only indication of its type.	Required input
businessDocumentIDRef	The XML IDREF version of businessDocument	Optional input
isAuthenticated	There is a digital certificate associated with the document entity. This provides proof of the signer's identity. (See also section on Document Security)	**false** {true, false}
isConfidential	The information entity is encrypted so that unauthorized parties cannot view the information (See also section on Document Security)	**false** {true, false}
isTamperProof	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)	**false** Valid values {true, false}
mimeType	Defines the valid MIME (Multipurpose Internet Mail Extensions) type of this Attachment	Optional input. Example: 'application/pdf'
specification	A reference to an external source of description of this attachment.	Optional Input
version	The version of the Attachment	Optional Input

2345

b. Authorized Role

XML Element Name: AuthorizedRole

DTD Declaration:

```
<!ELEMENT AuthorizedRole (Documentation*)>
<!ATTLIST AuthorizedRole
          name   CDATA #REQUIRED
          nameID ID    #IMPLIED>
```

Definition:

An Authorized Role is a role that is authorized to send the request or response, e.g. the buyer is authorized to send the request for purchase order, the seller is authorized to send the acceptance of purchase order.

.

Parents:

BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
Name	Defines the name of the Authorized Role	**Required input.**
nameID	XML ID version of name	Optional

c. Binary Collaboration

XML Element Name: BinaryCollaboration

DTD Declaration:

```
<!ELEMENT BinaryCollaboration (Documentation*,
AuthorizedRole, AuthorizedRole,(Documentation* | Start |
Transition | Success | Failure |
BusinessTransactionActivity | CollaborationActivity | Fork
| Join)*)>
<!ATTLIST BinaryCollaboration
      name          CDATA #REQUIRED
      nameID        ID    #IMPLIED
      pattern       CDATA #IMPLIED
      beginsWhen    CDATA #IMPLIED
      endsWhen      CDATA #IMPLIED
      precondition  CDATA #IMPLIED
      postCondition CDATA #IMPLIED
      timeToPerform CDATA #IMPLIED
>
```

Definition:

A Binary Collaboration defines a protocol of interaction between two authorized roles.

A 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.

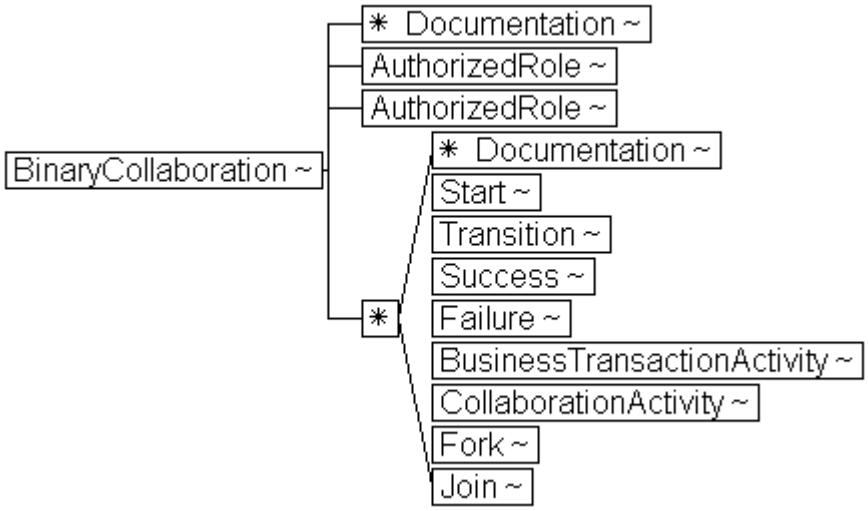
A Binary Collaboration choreographs one or more business transaction activities between two roles.

A Binary Collaboration is not an atomic transaction and should not be used in cases where Business Transaction rollback is required.

Parents:

- Package

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input.
nameID	The XML ID version of name	Optional
beginsWhen	A description of an event external to the collaboration that normally causes this collaboration to commence.	Optional Input.
endsWhen	A description of an event external to this collaboration that normally causes this collaboration to conclude.	Optional Input.
pattern	The optional reference to a pattern that this binary collaboration is based on.. In the XML Schema version the data type is xsd:anyURI	

preCondition	A description of a state external to this collaboration that is required before this collaboration can commence.	Optional Input.
postCondition	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.	Optional Input..
timeToPerform	The period of time, starting upon initiation of the first activity, within which this entire collaboration must conclude.	Optional Input.

c: Business Partner Role

Element Name: BusinessPartnerRole

DTD Declaration:

```
<!ELEMENT BusinessPartnerRole (Documentation*, Performs*,
                               Transition*)>
<!ATTLIST BusinessPartnerRole
          name   CDATA #REQUIRED
          nameID ID    #IMPLIED>
```

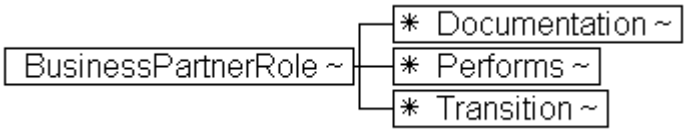
Definition:

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

Parents:

- MultiPartyCollaboration

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value
Name	Defines the name of the role played by partner in the overall multiparty business collaboration, e.g. customer or supplier.	Required Input.

nameID	The XML ID version of name	Optional

d. Business Transaction

Element Name: BusinessTransaction

Content Model:

```
<!ELEMENT BusinessTransaction (Documentation*,
RequestingBusinessActivity, RespondingBusinessActivity)>
<!ATTLIST BusinessTransaction
      name                          CDATA #REQUIRED
      nameID                        ID    #IMPLIED
      pattern                       CDATA #IMPLIED
      beginsWhen                    CDATA #IMPLIED
      endsWhen                      CDATA #IMPLIED
      isGuaranteedDeliveryRequired (true | false) false
      precondition                  CDATA #IMPLIED
      postCondition                 CDATA #IMPLIED>
```

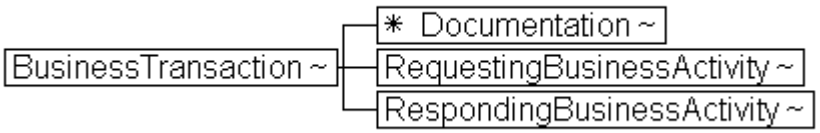
Definition:

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 an in the distribution of product announcements.

Parents:

- Package

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of the Business Transaction.	Required Input.

nameID	The XML ID version of name	Optional
pattern	The optional reference to a pattern that this transaction is based on.In the XML Schema version the data type is xsd:anyURI	Optional
beginsWhen	A description of an event external to the transaction that normally causes this transaction to commence.	Optional Input..
endsWhen	A description of an event external to this transaction that normally causes this transaction to conclude.	Optional Input.
isGuaranteedDeliveryRequired	Both partners must agree to use a transport that guarantees delivery	**false** Valid Values: {true, false}
preCondition	A description of a state external to this transaction that is required before this transaction can commence.	Optional Input.
postCondition	A description of a state that does not exist before the execution of this transaction but will exist as a result of the execution of this transaction.	Optional Input.

e. Business Transaction Activity

Element Name: BusinessTransactionActivity

Content Model:

```
<!ELEMENT BusinessTransactionActivity (Documentation*)>
<!ATTLIST BusinessTransactionActivity
          name                    CDATA #REQUIRED
          nameID                  ID    #IMPLIED
          businessTransaction     CDATA #REQUIRED
          businessTransactionIDRef IDREF #IMPLIED
          fromAuthorizedRole      CDATA #REQUIRED
          fromAuthorizedRoleIDRef  IDREF #IMPLIED
          toAuthorizedRole        CDATA #REQUIRED
          toAuthorizedRoleIDRef    IDREF #IMPLIED
          isConcurrent (true | false) "false"
          isLegallyBinding (true | false) "true"
          timeToPerform CDATA #IMPLIED>
```

Definition:

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.

Parents:

- BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of the activity uniquely within the binary collaboration	Required Input.
nameID	The XML ID version of name	Optional Input
businessTransaction	A reference, by name to the Business Transaction performed by this Business Transaction Activity	Required Input.
businessTransactionIDRef	The XML IDREF version of businessTransaction	Optional Input.
fromAuthorizedRole	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	Required Input.
fromAuthorizedRoleIDRef	The XML IDREF version of fromAuthorizedRole	OptionalInput.
toAuthorizedRole	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	Required Input.
toAuthorizedRoleIDRef	The XML IDREF version of toAuthorizedRole	Optional Input.
timeToPerform	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.	Optional Input.

isLegallyBinding	Defines whether the Business Transaction performed by this activity is intended by the trading parties to be binding. Default value is True.	**true** Valid Values: {true, false}
isConcurrent	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	**false** Valid Values: {true, false}

f. Collaboration Activity

Element Name: CollaborationActivity

DTD Declaration:

```
<!ELEMENT CollaborationActivity (Documentation*)>
<!ATTLIST CollaborationActivity
      name                    CDATA #REQUIRED
      nameID                  ID    #IMPLIED
      fromAuthorizedRole      CDATA #REQUIRED
      fromAuthorizedRoleIDRef CDATA #IMPLIED
      toAuthorizedRole        CDATA #REQUIRED
      toAuthorizedRoleIDRef   CDATA #IMPLIED
      binaryCollaboration     CDATA #REQUIRED>
      binaryCollaborationIDRef CDATA #IMPLIED>
```

Definition:

A collaboration activity is the activity of performing a binary collaboration within another binary collaboration.

Parents:

- BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of the activity uniquely within the binary collaboration	Required Input.
fromAuthorizedRole	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	Required Input

fromAuthorizedRoleIDRef	The XML IDREF version of fromAuthorizedRole	OptionalInput.
toAuthorizedRole	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 respondier in the BinaryCollaboration performed by this activity	Required Input.
toAuthorizedRoleIDRef	The XML IDREF version of toAuthorizedRole	Optional Input.
binaryCollaboration	A reference, by name, to the Binary Collaboration performed by this Collaboration Activity	Required Input.
binaryCollaborationIDRef	The XML IDREF version of binaryCollaboration	Optional Input.

g. Documentation

Element Name: Documentation

DTD Declaration:

```
<!ELEMENT Documentation (#PCDATA)>
<!ATTLIST Documentation
      uri CDATA #IMPLIED>
```

Definition:

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

Parents:

- `AuthorizedRole`
- `BinaryCollaboration`
- `BusinessPartnerRole`
- `BusinessTransaction`
- `BusinessTransactionActivity`
- `CollaborationActivity`
- `DocumentEnvelope`
- `BusinessDocument`
- `ProcessSpecification`
- `MultiPartyCollaboration`
- `Package`
- `Performs`
- `RequestingBusinessActivity`

- RespondingBusinessActivity
- DocumentSpecification
- Transition

Attributes:

Attribute Name	Definition	Default Value
uri	Defines the URI (Uniform Resource Identifier) where external documentation is located. In the XML Schema version the data type is xsd:anyURI	No Default Value. Valid URI is required.

h. DocumentEnvelope

Element Name: DocumentEnvelope

Content Model:

```
<!ELEMENT DocumentEnvelope (Documentation*,
BusinessDocument,
                        Attachment*)>
<!ATTLIST DocumentEnvelope
      isPositiveResponse CDATA #IMPLIED
      isAuthenticated   (true | false) "false"
      isConfidential    (true | false) "false"
      isTamperProof     (true | false) "false">
```

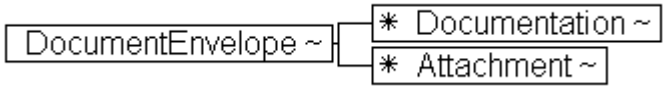
Definition:

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.

Parents:

- RequestingBusinessActivity
- RespondingBusinessActivity

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value

isPositiveResponse	An expression whose evaluation results in 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.	Optional Input.
isAuthenticated	There is a digital certificate associated with the document entity. This provides proof of the signer's identity. (See also section on Document Security)	**false** Valid Values: {true, false}
isConfidential	The information entity is encrypted so that unauthorized parties cannot view the information. (See also section on Document Security)	**false** Valid Values: {true, false}
isTamperProof	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)	**false** Valid Values: {true, false}

2563
2564
2565 **i. BusinessDocument**
2566 **Element Name:** BusinessDocument

DTD Declaration:

```
<!ELEMENT BusinessDocument  (Documentation*) >
<!ATTLIST BusinessDocument
      name   CDATA #REQUIRED
      nameID ID    #IMPLIED  >
```

Definition:

BusinessDocument is a generic name of a document. The location of the definition of the document can be found in the associated DocumentSpecification.

Parents:

- Attachment
- DocumentSpecification

Attributes:

Attribute Name	Definition	Default Value
name	Defines the generic name of the Business Document as it is known within this Business Process Specification	Required Input
nameID	XML ID version of name	Optional

j. DocumentSpecification

Element Name:

DTD Declaration:

```
<!ELEMENT DocumentSpecification (Documentation*,
BusinessDocument)>
<!ATTLIST DocumentSpecification
      name         CDATA #REQUIRED
      nameID       ID    #IMPLIED
      location     CDATA #IMPLIED
      logicalModel CDATA #IMPLIED
      version      CDATA #IMPLIED>
```

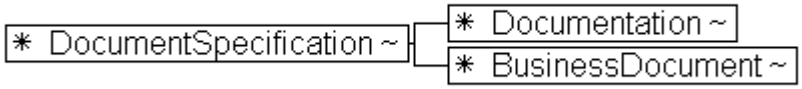
Definition:

A DocumentSpecification is a collection of Document Definitions. The DocumentSpecification is usually external to the process specification, and is referenced with a URI. An additional reference is to where the logical model is for the Documents in the DocumentSpecification. Typically this would be an ebXML core component context model.

Parents:

- Package

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value
name	Defines the generic name of the DocumentSpecification as it is known within this Business Process Specification	Required
nameID	XML ID version of name	Optional
location	Reference to an external source of the schema definition. In the XML Schema version the data type is xsd:anyURI	No default value.
logicalModel	Reference is to where the logical model is. In the XML Schema version the data type is xsd:anyURI	No default value.
version	The version of the document specification	Optional

k. Failure

Element Name: Failure

DTD Declaration:

```
<!ELEMENT Failure   (Documentation*)   >
<!ATTLIST Failure
          fromBusinessState      CDATA #REQUIRED
          fromBusinessStateIDRef IDREF #IMPLIED
          guardCondition (Success | BusinessFailure |
                 TechnicalFailure | AnyFailure) #IMPLIED
          guardExpression    CDATA    #IMPLIED>
```

Definition:

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

Parents:

- BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
fromBusinessState	The name of the activity from which this indicates a transition to unsuccessful conclusion of the BusinessTransaction or BinaryCollaboration	Required Input.
fromBusinessState IDRef	The XML IDREF version of fromBusinessState	Optional
guardCondition	The condition that guards this transition	Optional Valid Values: {Success, BusinessFailure, TechnicalFailure, AnyFailure}
guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition should happen or not. The expression can refer to the name or content of the the most recent DocumentEnvelope or content of documents within it.	Optional

l. Fork

Element Name: Fork

DTD Declaration:

```
<!ELEMENT Fork (Documentation*)   >
<!ATTLIST Fork
          name   CDATA  #REQUIRED
          nameID ID     #IMPLIED >
```

Definition:

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.

Parents:

- BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of the Fork state	Required Input
nameID	The XML ID version of name	Optional

m. Include

Element Name: Include

DTD Declaration:

```
<!ELEMENT Include   (Documentation*)   >
<!ATTLIST Include
          name      CDATA    #REQUIRED
          version   CDATA    #REQUIRED
          uuid      CDATA    #REQUIRED
          uri       CDATA    #REQUIRED  >
```

Definition:

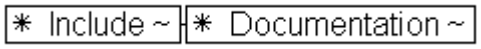
Includes another process specification document and merges that specification with the current specification. Any elements of the same name and in the same name scope must have exactly the same specification except that packages may have additional content.

Documents are merged based on name scope. A name in an included package will be indistinguishable from a name in the base document.

Parents:

- ProcessSpecification

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required
uri	Uniform Resource Indicator. In the XML Schema data type is xsd:anyURI	Required
uuid	Universally unique identifier.	Required
version	Version of the included specification.	Required

n. Join

Element Name: Join

DTD Declaration:

```
<!ELEMENT Join   (Documentation*)   >
<!ATTLIST Join
          name       CDATA #REQUIRED
          nameID     ID    #IMPLIED
          waitForAll (true | false) "true">
```

Definition:

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.

Parents:

- BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of the Join state.	Required Input
nameID	The XML ID version of name	Optional
waitForAll	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.	**true** Valid Values: {true, false}

o. MultiParty Collaboration

Element Name: MultiPartyCollaboration

DTD Declaration:

```
<!ELEMENT MultiPartyCollaboration  (Documentation*,
                                    BusinessPartnerRole+) >
<!ATTLIST MultiPartyCollaboration
          name    CDATA #REQUIRED
          nameID  ID    #IMPLIED  >
```

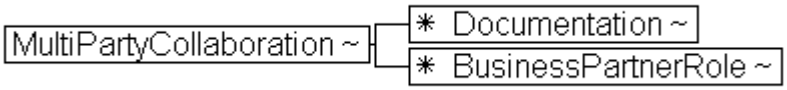
Definition:

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.

Parents:

- Package

Hierarchical Model:



\Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of the MultiPartyCollaboration	Required Input
nameID	The XML ID version of name	Optional Input

p. Package

Element Name: Package

DTD Declaration:

```
<!ELEMENT Package  (Documentation*, (Package |
                    BinaryCollaboration |
                    MultiPartyCollaboration |
                    BusinessTransaction)*) >
<!ATTLIST Package
      name   CDATA #REQUIRED
      nameID ID    #IMPLIED  >
```

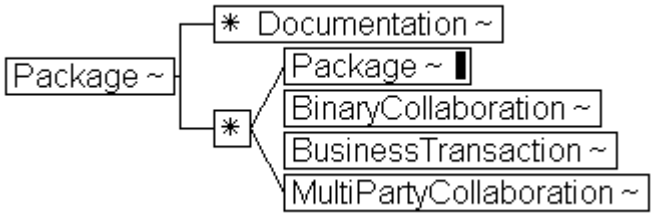
Definition:

Defines a hierarchical name scope containing reusable elements.

Parents:

- ProcessSpecification
- Package

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model.	Required Input
nameID	XML ID version of name	Optional

q. Performs

Element Name: Performs

DTD Declaration:

```
<!ELEMENT Performs  (Documentation*) >
<!ATTLIST Performs
          authorizedRole      CDATA    #REQUIRED
          authorizedRoleIDRef IDREF    #IMPLIED  >
```

Definition:

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.

Parents:

- BusinessPartnerRole

Attributes:

Attribute Name	Definition	Default Value
authorizedRole	The AuthorizedRole that will be performed by the Business PartnerRole, qualified with the name of the BinaryCollaboration	Required Input
authorizedRoleIDRef	The XML IDREF version of AuthorizedRole	Optional Input

r. ProcessSpecification

Element Name: ProcessSpecification

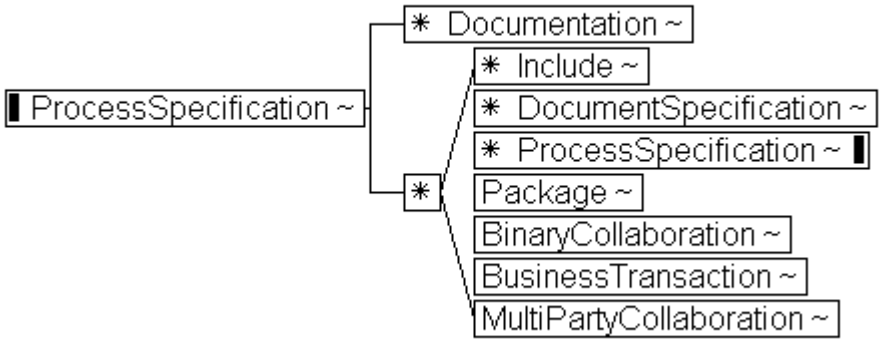
DTD Declaration:

```
<!ELEMENT ProcessSpecification (Documentation*, (Include* |
DocumentSpecification* | ProcessSpecification* | Package |
BinaryCollaboration | BusinessTransaction |
MultiPartyCollaboration)*)>
<!ATTLIST ProcessSpecification
          name         ID    #REQUIRED
          version      CDATA    #REQUIRED
          uuid         CDATA    #REQUIRED  >
```

Definition:

Root element of a process specification document that has a globally unique identity.

Hierarchical Model:



Attributes:

Attribute Name	Definition	Default Value
name	Defines the name of a model element. This name must be unique within the context of the model element and will be used to reference the element from other points in the model. It is defined as an XML ID.	Required
uuid	Universally unique identifier.	Required
version	Version of the specification.	Required

s. Requesting Business Activity

Element Name: RequestingBusinessActivity

DTD Declaration:

```
<!ELEMENT RequestingBusinessActivity  (Documentation*,
                                       DocumentEnvelope) >
<!ATTLIST RequestingBusinessActivity
    name                            CDATA         #IMPLIED
    nameID                          ID            #IMPLIED
    isAuthorizationRequired         (true | false) “false”
    isIntelligibleCheckRequired     (true | false) “false”
    isNonRepudiationReceiptRequired (true | false) “false”
    isNonRepudiationRequired        (true | false) “false”
    timeToAcknowledgeAcceptance      CDATA        #IMPLIED
    timeToAcknowledgeReceipt         CDATA        #IMPLIED>
```

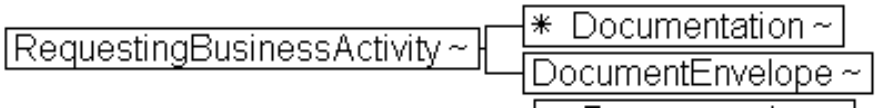
Definition:

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.

Parents:

- BusinessTransaction

2778 **Hierarchical Model:**



2779

2780 **Attributes:**

Attribute Name	Definition	Default Value
name	Defines the name of the RequestingBusinessTransaction	Optional Input
nameID	The XML ID version of name	Optional Input
isAuthorizationRequired	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)	**false** Valid Values: {true, false}
isIntelligibleCheckRequired	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)	**false** Valid Values: {true, false}
isNonRepudiationReceiptRequired	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)	**false** Valid Values: {true, false}
isNonRepudiationRequired	Requires the sending parties to save copies of the transacted documents before sending them (See also section on nonrepuditation)	**false** Valid Values: {true, false}

timeToAcknowledgeAcceptance	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)	No default value.
timeToAcknowledgeReceipt	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)	No default value.

t. Responding Business Activity

Element Name: RespondingBusinessActivity

DTD Declaration:

```
<!ELEMENT RespondingBusinessActivity  (Documentation*,
                                       DocumentEnvelope) >
<!ATTLIST RespondingBusinessActivity
    name                            CDATA         #IMPLIED
    nameID                          ID            #IMPLIED
    isAuthorizationRequired         (true | false) “false”
    isIntelligibleCheckRequired     (true | false) “false”
    isNonRepudiationReceiptRequired (true | false) “false”
    isNonRepudiationRequired        (true | false) “false”
    timeToAcknowledgeReceipt         CDATA        #IMPLIED>
```

Definition:

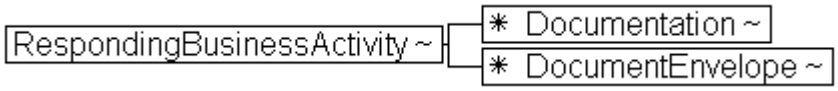
A RespondingBusinessActivity is a Business Action that is performed by the responding role within a Business Transaction. It specifies the Document Envelope which will carry the response.

There may be multiple possible response Document Envelopes defined, but only one of them will be sent during an actual transaction instance.

Parents:

- BusinessTransaction

Hierarchical Model:



2806
2807 **Attributes:**

Attribute Name	Definition	Default Value
name	Defines the name of the RespondingBusinessTransaction	Optional Input
nameID	The XML ID version of name	Optional Input
isAuthorizationRequired	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)	**false** Valid Values: {true, false}
isIntelligibleCheckRequired	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)	**false** Valid Values: {true, false}
isNonRepudiationReceiptRequired	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)	**false** Valid Values: {true, false}
isNonRepudiationRequired	Requires the sending parties to save copies of the transacted documents before sending them (See also section on nonrepuditation)	**false** Valid Values: {true, false}
timeToAcknowledgeReceip	The time the receiving party has to acknowledge receipt	No default

t	of a business document. This parameter is specified on the sending side. (See also section on core transaction semantics)	value.

u. Start

Element Name: Start

DTD Declaration:

```
<!ELEMENT Start   (Documentation*)   >
<!ATTLIST Start
          toBusinessState      CDATA #REQUIRED
          toBusinessStateIDRef IDREF #IMPLIED  >
```

Definition:

The starting state for an Binary Collaboration. A Binary Collaboration should have at least one starting activity. If none defined, then all activities are considered allowable entry points.

Parents:

- BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
toBusinessState	The name of an activity which an allowable starting point for this for BinaryCollaboration	Required Input
toBusinessStateIDRef	The XML IDREF version of toBusinessState	Optional

v. Success

Element Name: Success

DTD Declaration:

```
<!ELEMENT Success   (Documentation*)   >
<!ATTLIST Success
          fromBusinessState  CDATA    #REQUIRED
          guardCondition (Success | BusinessFailure |
                 TechnicalFailure | AnyFailure) #IMPLIED
          guardExpression    CDATA    #IMPLIED>
```

Definition:

Defines the successful conclusion of a binary collaboration as a transition from an activity.

Parents:

- BinaryCollaboration

Attributes:

Attribute Name	Definition	Default Value
fromBusinessState	The name of the activity from which this indicates a transition to successful conclusion of the BusinessTransaction or BinaryCollaboration	Required Input.
guardCondition	The condition that guards this transition	Optional Valid Values: {Success, BusinessFailure, TechnicalFailure, AnyFailure}
guardExpression	An expression whose evaluation results in TRUE or FALSE as the determination of whether this DocumentEnvelope should be considered the successful initiation or conclusion of a BusinessTransaction. In some situations this could be an XPath expression that interrogates the BusinessDocument.	Optional

w. Transition

ELEMENT Name: Transition

DTD Declaration:

```
<!ELEMENT Transition  (Documentation*) >
<!ATTLIST Transition
          onInitiation (true | false)  "false"
          fromBusinessState      CDATA #IMPLIED
          fromBusinessStateIDRef IDREF #IMPLIED
          toBusinessState        CDATA #IMPLIED
          toBusinessStateIDRef   IDREF #IMPLIED
          guardCondition (Success | BusinessFailure |
                 TechnicalFailure | AnyFailure) #IMPLIED
          guardExpression CDATA #IMPLIED>
```

Definition:

A transition is a transition between two business states in a binary collaboration.

Choreography is expressed as transitions between business states.

Parents:

- BinaryCollaboration
- BusinessPartnerRole

Attributes:

Attribute Name	Definition	Default Value
onInitiation	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	**false** Valid Values: {true, false}
fromBusinessState	The name of the state transitioned from	No default value.
fromBusinessStateIDRef	The XML IDREF version of fromBusinessState	Optional
toBusinessState	The name of the state transitioned to	No default value.
toBusinessStateIDRef	The XML IDREF version of toBusinessState	Optional

guardCondition	A reference to the status of the previous transaction. A fixed value of Success, BusinessFailure, TechnicalFailure, or AnyFailure	`Optional`
		`Valid Values:`
		`{Success,`
		`BusinessFailure,`
		`TechnicalFailure,`
		`AnyFailure}`

guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition should happen or not. The expression can refer to the name or content of the the most recent DocumentEnvelope or content of documents within it. In some situations this could be an XPath expression that interrogates the BusinessDocument.	`Optional`

8.2 XML to UML cross-reference

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

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

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

- BusinessState
- CompletionState
- BusinessActivity
- BusinessAction
- DocumentSecurity

The following classes in the UML specification schema are in the DTD represented not by elements but by attributes in other elements:

- BusinessDocument (attribute of DocumentEnvelope and of Schema)

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 ebXMLProcessSpecification can be arranged in a recursive package structure. The ebXMLProcessSpecification 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 PackageContent are BusinessTransaction, BinaryCollaboration, MultiPartyCollaboration.

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

Each PackageContent 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 is 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 DTD or 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">
        <AuthorizedRole name="buyer" nameID="r224"/>
        <AuthorizedRole name="seller" nameID="r225"/>
    </BinaryCollaboration>
</Package>

<!-the XPath approach -->
<Performs
authorizedRole='//Package[@name="OAGOrdering"]/BinaryCollaboration[@name="OrderColla
    boration"]/AuthorizedRole[@name="buyer"]'/>

<!-Combination approach -->
<Performs authorizedRole="buyer" authorizedRoleIDRef="r222"/>
```

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 authorizedRole='//AuthorizedRole[@name="buyer"]'/>

Relative paths are also allowed for example:

<BusinessTransactionActivity fromAuthorizedRole='../AuthorizedRole[@name="buyer"]' ... />

8.4 Sample XML document against above DTD

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 DTD's for business signal payload for receiptAcknowledgment (from the RosettaNet website, courtesy of RosettaNet, and Edifecs) and for acceptanceAcknowledgement and exception.

9.1.1 ReceiptAcknowledgment DTD

```
<!--

   RosettaNet XML Message Schema.
   http://www.rosettanet.org
   RosettaNet XML Message Schema.
   Receipt Acknowledgement
   Version 1.1
-->

<!ENTITY % common-attributes "id  CDATA #IMPLIED">

<!ELEMENT ReceiptAcknowledgement (
          fromRole ,
          NonRepudiationInformation? ,
          receivedDocumentDateTime ,
          receivedDocumentIdentifier ,
          thisMessageDateTime ,
          thisMessageIdentifier ,
```

```
                    toRole ) >

          <!ELEMENT fromRole
                      ( PartnerRoleDescription ) >

          <!ELEMENT PartnerRoleDescription (
                    ContactInformation? ,
                    GlobalPartnerRoleClassificationCode ,
                    PartnerDescription ) >

          <!ELEMENT ContactInformation (
                    contactName ,
                    EmailAddress ,
                    telephoneNumber ) >

          <!ELEMENT contactName
                     ( FreeFormText ) >

          <!ELEMENT FreeFormText
                     ( #PCDATA ) >
          <!ATTLIST FreeFormText
                     xml:lang CDATA #IMPLIED >

          <!ELEMENT EmailAddress
                     ( #PCDATA ) >

          <!ELEMENT telephoneNumber
                     ( CommunicationsNumber ) >

          <!ELEMENT CommunicationsNumber
                     ( #PCDATA ) >

          <!ELEMENT GlobalPartnerRoleClassificationCode
                     ( #PCDATA ) >

          <!ELEMENT PartnerDescription (
                    BusinessDescription ,
                    GlobalPartnerClassificationCode ) >

          <!ELEMENT BusinessDescription (
                    GlobalBusinessIdentifier ,
                    GlobalSupplyChainCode ) >

          <!ELEMENT GlobalBusinessIdentifier
                     ( #PCDATA ) >

          <!ELEMENT GlobalSupplyChainCode
                     ( #PCDATA ) >

          <!ELEMENT GlobalPartnerClassificationCode
                     ( #PCDATA ) >

          <!ELEMENT NonRepudiationInformation (
                    GlobalDigestAlgorithmCode ,
                    OriginalMessageDigest ) >
```

```
<!ELEMENT GlobalDigestAlgorithmCode
          ( #PCDATA ) >

<!ELEMENT OriginalMessageDigest
          ( #PCDATA ) >

<!ELEMENT receivedDocumentDateTime
          ( DateTimeStamp ) >

<!ELEMENT DateTimeStamp
          ( #PCDATA ) >

<!ELEMENT receivedDocumentIdentifier
          ( ProprietaryDocumentIdentifier ) >

<!ELEMENT ProprietaryDocumentIdentifier
          ( #PCDATA ) >

<!ELEMENT thisMessageDateTime
           ( DateTimeStamp ) >

<!ELEMENT thisMessageIdentifier
          ( ProprietaryMessageIdentifier ) >

<!ELEMENT ProprietaryMessageIdentifier
          ( #PCDATA ) >

<!ELEMENT toRole
           ( PartnerRoleDescription ) >
```

9.1.2 AcceptanceAcknowledgement DTD

```
<!--
   RosettaNet XML Message Schema.
   http://www.rosettanet.org
   RosettaNet XML Message Schema.
   Acceptance Acknowledgement Exception
   Version 1.1
-->

<!ENTITY % common-attributes "id  CDATA #IMPLIED">

<!ELEMENT AcceptanceAcknowledgementException (
          fromRole ,
          reason ,
          theMessageDatetime ,
          theOffendingDocumentDateTime ,
          theOffendingDocumentIdentifier ,
          thisMessageIdentifier ,
          toRole ) >

<!ELEMENT fromRole
           ( PartnerRoleDescription ) >
```

```
<!ELEMENT PartnerRoleDescription (
          ContactInformation? ,
          GlobalPartnerRoleClassificationCode ,
          PartnerDescription ) >

<!ELEMENT ContactInformation (
          contactName ,
          EmailAddress ,
          telephoneNumber ) >

<!ELEMENT contactName
           ( FreeFormText ) >

<!ELEMENT FreeFormText
           ( #PCDATA ) >
<!ATTLIST FreeFormText
           xml:lang CDATA #IMPLIED >

<!ELEMENT EmailAddress
           ( #PCDATA ) >

<!ELEMENT telephoneNumber
           ( CommunicationsNumber ) >

<!ELEMENT CommunicationsNumber
           ( #PCDATA ) >

<!ELEMENT GlobalPartnerRoleClassificationCode
           ( #PCDATA ) >

<!ELEMENT PartnerDescription (
          BusinessDescription ,
          GlobalPartnerClassificationCode ) >

<!ELEMENT BusinessDescription (
          GlobalBusinessIdentifier ,
          GlobalSupplyChainCode ) >

<!ELEMENT GlobalBusinessIdentifier
           ( #PCDATA ) >

<!ELEMENT GlobalSupplyChainCode
           ( #PCDATA ) >

<!ELEMENT GlobalPartnerClassificationCode
           ( #PCDATA ) >

<!ELEMENT reason
            ( FreeFormText ) >

<!ELEMENT theMessageDatetime
           ( DateTimeStamp ) >

<!ELEMENT DateTimeStamp
           ( #PCDATA ) >
```

```
<!ELEMENT theOffendingDocumentDateTime
            ( DateTimeStamp ) >

<!ELEMENT theOffendingDocumentIdentifier
           ( ProprietaryDocumentIdentifier ) >

<!ELEMENT ProprietaryDocumentIdentifier
           ( #PCDATA ) >

<!ELEMENT thisMessageIdentifier
           ( ProprietaryMessageIdentifier ) >

<!ELEMENT ProprietaryMessageIdentifier
           ( #PCDATA ) >

<!ELEMENT toRole
            ( PartnerRoleDescription ) >
```

9.1.3 Exception Signal DTD

```
<!--
   RosettaNet XML Message Schema.
   http://www.rosettanet.org
   RosettaNet XML Message Schema.
   Exception
   Version 1.1
-->


<!ENTITY % common-attributes "id  CDATA #IMPLIED">

<!ELEMENT Exception (
          fromRole? ,
          reason ,
          theMessageDatetime ,
          theOffendingDocumentDateTime? ,
          theOffendingDocumentIdentifier? ,
          thisMessageIdentifier ,
          toRole? ) >

<!ELEMENT fromRole
            ( PartnerRoleDescription ) >

<!ELEMENT PartnerRoleDescription (
          ContactInformation? ,
          GlobalPartnerRoleClassificationCode? ,
          PartnerDescription? ) >

<!ELEMENT ContactInformation (
          contactName? ,
          EmailAddress? ,
          telephoneNumber? ) >
```

```
<!ELEMENT contactName
          ( FreeFormText ) >

<!ELEMENT FreeFormText
          ( #PCDATA ) >
<!ATTLIST FreeFormText
          xml:lang CDATA #IMPLIED >

<!ELEMENT EmailAddress
          ( #PCDATA ) >

<!ELEMENT telephoneNumber
          ( CommunicationsNumber ) >

<!ELEMENT CommunicationsNumber
          ( #PCDATA ) >

<!ELEMENT GlobalPartnerRoleClassificationCode
          ( #PCDATA ) >

<!ELEMENT PartnerDescription (
          BusinessDescription? ,
          GlobalPartnerClassificationCode? ) >

<!ELEMENT BusinessDescription (
          GlobalBusinessIdentifier? ,
          GlobalSupplyChainCode? ) >

<!ELEMENT GlobalBusinessIdentifier
          ( #PCDATA ) >

<!ELEMENT GlobalSupplyChainCode
          ( #PCDATA ) >

<!ELEMENT GlobalPartnerClassificationCode
          ( #PCDATA ) >

<!ELEMENT reason
           ( FreeFormText ) >

<!ELEMENT theMessageDatetime
          ( DateTimeStamp ) >

<!ELEMENT DateTimeStamp
          ( #PCDATA ) >

<!ELEMENT theOffendingDocumentDateTime
           ( DateTimeStamp ) >

<!ELEMENT theOffendingDocumentIdentifier
          ( ProprietaryDocumentIdentifier ) >

<!ELEMENT ProprietaryDocumentIdentifier
          ( #PCDATA ) >
```

```
<!ELEMENT thisMessageIdentifier
          ( ProprietaryMessageIdentifier ) >

<!ELEMENT ProprietaryMessageIdentifier
          ( #PCDATA ) >

<!ELEMENT toRole
           ( PartnerRoleDescription ) >
```

10 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 DTD version 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 the DTD. 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 DTD. 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 the DTD. 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

```
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Kurt
Kanaskie (Lucent Technologies) -->
<!-- Notes from Kurt Kanaskie on 2001-04-17
     Differences from 099 DTD:
      binaryCollaboration attribute in Performs
      fromBinaryCollaboration attribute in Transition
      toBinaryCollaboration attribute in Transition
     Differences from original:
      guard instead of guardExpression
     See EBXMLSpecification_2001_04_23.dtd for other changes
-->
<!DOCTYPE ProcessSpecification SYSTEM "ebXMLProcessSpecification-
v1.00.dtd">
<ProcessSpecification name="Simple" version="1.1" uuid="[1234-5678-
901234]">
      <!-- Business Documents -->
      <DocumentSpecification name="EBXML">
            <BusinessDocument name="Catalog Request"/>
            <BusinessDocument name="Catalog"/>
            <BusinessDocument name="Purchase Order"/>
            <BusinessDocument name="PO Acknowledgement"/>
            <BusinessDocument name="Credit Request"/>
            <BusinessDocument name="Credit Confirm"/>
            <BusinessDocument name="ASN"/>
            <BusinessDocument name="CreditAdvice"/>
            <BusinessDocument name="DebitAdvice"/>
            <BusinessDocument name="Invoice"/>
            <BusinessDocument name="Payment"/>
            <BusinessDocument name="Inventory Report Request"/>
            <BusinessDocument name="Inventory Report"/>
            <BusinessDocument name="Inventory Report"/>
      </DocumentSpecification>
      <Package name="Ordering">
            <!-- First the overall MultiParty Collaboration -->
            <MultiPartyCollaboration name="DropShip">
                  <BusinessPartnerRole name="Customer">
                        <Performs authorizedRole="requestor"/>
                        <Performs authorizedRole="buyer"/>
                        <Transition fromBusinessState="Catalog Request"
toBusinessState="Create Order"/>
                  </BusinessPartnerRole>
                  <BusinessPartnerRole name="Retailer">
                        <Performs authorizedRole="provider"/>
                        <Performs authorizedRole="seller"/>
                        <Performs authorizedRole="Creditor"/>
                        <Performs authorizedRole="buyer"/>
                        <Performs authorizedRole="Payee"/>
                        <Performs authorizedRole="Payor"/>
                        <Performs authorizedRole="requestor"/>
```

```
                        <Transition fromBusinessState="Create Order"
toBusinessState="Check Credit"/>
                        <Transition fromBusinessState="Check Credit"
toBusinessState="Create Order"/>
                  </BusinessPartnerRole>
                  <BusinessPartnerRole name="DropShip Vendor">
                        <Performs authorizedRole="seller"/>
                        <Performs authorizedRole="payee"/>
                        <Performs authorizedRole="provider"/>
                  </BusinessPartnerRole>
                  <BusinessPartnerRole name="Credit Authority">
                        <Performs authorizedRole="credit service"/>
                        <Performs authorizedRole="payor"/>
                  </BusinessPartnerRole>
            </MultiPartyCollaboration>
            <!-- Now the Binary Collaborations -->
            <BinaryCollaboration name="Request Catalog">
                  <AuthorizedRole name="requestor"/>
                  <AuthorizedRole name="provider"/>
                  <BusinessTransactionActivity name="Catalog Request"
businessTransaction="Catalog Request" fromAuthorizedRole="requestor"
toAuthorizedRole="provider"/>
            </BinaryCollaboration>
            <BinaryCollaboration name="Firm Order" timeToPerform="P2D">
                  <Documentation>timeToPerform = Period: 2 days from
start of transaction</Documentation>
                  <AuthorizedRole name="buyer"/>
                  <AuthorizedRole name="seller"/>
                  <BusinessTransactionActivity name="Create Order"
businessTransaction="Create Order" fromAuthorizedRole="buyer"
toAuthorizedRole="seller"/>
            </BinaryCollaboration>
            <BinaryCollaboration name="Product Fulfillment"
timeToPerform="P5D">
                  <Documentation>timeToPerform = Period: 5 days from
start of transaction</Documentation>
                  <AuthorizedRole name="buyer"/>
                  <AuthorizedRole name="seller"/>
                  <BusinessTransactionActivity name="Create Order"
businessTransaction="Create Order" fromAuthorizedRole="buyer"
toAuthorizedRole="seller"/>
                  <BusinessTransactionActivity name="Notify shipment"
businessTransaction="Notify of advance shipment"
fromAuthorizedRole="buyer" toAuthorizedRole="seller"/>
                  <Start toBusinessState="Create Order"/>
                  <Transition fromBusinessState="Create Order"
toBusinessState="Notify shipment"/>
                  <Success fromBusinessState="Notify shipment"
guardExpression="Success"/>
                  <Failure fromBusinessState="Notify shipment"
guardExpression="BusinessFailure"/>
            </BinaryCollaboration>
            <BinaryCollaboration name="Inventory Status">
                  <AuthorizedRole name="requestor"/>
                  <AuthorizedRole name="provider"/>
```

```
                  <BusinessTransactionActivity name="Inventory Report
Request" businessTransaction="Inventory Report Request"
fromAuthorizedRole="requestor" toAuthorizedRole="provider"/>
                  <BusinessTransactionActivity name="Inventory Report"
businessTransaction="Inventory Report" fromAuthorizedRole="provider"
toAuthorizedRole="requestor"/>
            </BinaryCollaboration>
            <BinaryCollaboration name="Credit Inquiry">
                  <AuthorizedRole name="creditor"/>
                  <AuthorizedRole name="credit service"/>
                  <BusinessTransactionActivity name="Check Credit"
businessTransaction="Check Credit" fromAuthorizedRole="creditor"
toAuthorizedRole="credit service"/>
            </BinaryCollaboration>
            <BinaryCollaboration name="Credit Payment">
                  <AuthorizedRole name="payee"/>
                  <AuthorizedRole name="payor"/>
                  <BusinessTransactionActivity name="Process Credit
Payment" businessTransaction="Process Credit Payment"
fromAuthorizedRole="payee" toAuthorizedRole="payor"/>
            </BinaryCollaboration>
            <!-- A compound BinaryCollaboration for illustration
purposes-->
            <BinaryCollaboration name="Credit Charge">
                  <AuthorizedRole name="charger"/>
                  <AuthorizedRole name="credit service"/>
                  <CollaborationActivity name="Credit Inquiry"
binaryCollaboration="Credit Inquiry" fromAuthorizedRole="creditor"
toAuthorizedRole="credit service"/>
                  <CollaborationActivity name="Credit Payment"
binaryCollaboration="Credit Payment" fromAuthorizedRole="payee"
toAuthorizedRole="payor"/>
                  <Transition fromBusinessState="Credit Inquiry"
toBusinessState="Credit Payment"/>
            </BinaryCollaboration>
            <BinaryCollaboration name="Fulfillment Payment">
                  <AuthorizedRole name="payee"/>
                  <AuthorizedRole name="payor"/>
                  <BusinessTransactionActivity name="Process Payment"
businessTransaction="Process Payment" fromAuthorizedRole="payee"
toAuthorizedRole="payor"/>
            </BinaryCollaboration>
            <!-- Here are all the Business Transactions needed -->
            <BusinessTransaction name="Catalog Request">
                  <RequestingBusinessActivity name="">
                        <DocumentEnvelope isPositiveResponse="true"
businessDocument="Catalog Request"/>
                  </RequestingBusinessActivity>
                  <RespondingBusinessActivity name="">
                        <DocumentEnvelope isPositiveResponse="true"
businessDocument="Catalog"/>
                  </RespondingBusinessActivity>
            </BusinessTransaction>
            <BusinessTransaction name="Create Order">
```

```
3480                    <RequestingBusinessActivity name=""
3481    isNonRepudiationRequired="true" timeToAcknowledgeReceipt="P2D"
3482    timeToAcknowledgeAcceptance="P3D">
3483                            <DocumentEnvelope isPositiveResponse="true"
3484    businessDocument="Purchase Order"/>
3485                    </RequestingBusinessActivity>
3486                    <RespondingBusinessActivity name=""
3487    isNonRepudiationRequired="true" timeToAcknowledgeReceipt="P5D">
3488                            <DocumentEnvelope isPositiveResponse="true"
3489    businessDocument="PO Acknowledgement"/>
3490                    </RespondingBusinessActivity>
3491             </BusinessTransaction>
3492             <BusinessTransaction name="Check Credit ">
3493                    <RequestingBusinessActivity name="">
3494                            <DocumentEnvelope isPositiveResponse="true"
3495    businessDocument="Credit Request"/>
3496                    </RequestingBusinessActivity>
3497                    <RespondingBusinessActivity name="">
3498                            <DocumentEnvelope isPositiveResponse="true"
3499    businessDocument="Credit Confirm"/>
3500                    </RespondingBusinessActivity>
3501             </BusinessTransaction>
3502             <BusinessTransaction name="Notify of advance shipment">
3503                    <RequestingBusinessActivity name="">
3504                            <DocumentEnvelope isPositiveResponse="true"
3505    businessDocument="ASN"/>
3506                    </RequestingBusinessActivity>
3507                    <RespondingBusinessActivity name=""
3508    timeToAcknowledgeReceipt="P2D"/>
3509             </BusinessTransaction>
3510             <BusinessTransaction name="Process Credit Payment">
3511                    <RequestingBusinessActivity name="">
3512                            <DocumentEnvelope isPositiveResponse="true"
3513    businessDocument="CreditAdvice"/>
3514                    </RequestingBusinessActivity>
3515                    <RespondingBusinessActivity name="">
3516                            <DocumentEnvelope isPositiveResponse="true"
3517    businessDocument="DebitAdvice"/>
3518                    </RespondingBusinessActivity>
3519             </BusinessTransaction>
3520             <BusinessTransaction name="Process Payment">
3521                    <RequestingBusinessActivity name="">
3522                            <DocumentEnvelope isPositiveResponse="true"
3523    businessDocument="Invoice"/>
3524                    </RequestingBusinessActivity>
3525                    <RespondingBusinessActivity name="">
3526                            <DocumentEnvelope isPositiveResponse="true"
3527    businessDocument="Payment"/>
3528                    </RespondingBusinessActivity>
3529             </BusinessTransaction>
3530             <BusinessTransaction name="Request Inventory Report">
3531                    <RequestingBusinessActivity name="">
3532                            <DocumentEnvelope isPositiveResponse="true"
3533    businessDocument="Inventory Report Request"/>
3534                    </RequestingBusinessActivity>
```

```
                  <RespondingBusinessActivity name="">
                        <DocumentEnvelope isPositiveResponse="true"
businessDocument="Inventory Report"/>
                  </RespondingBusinessActivity>
            </BusinessTransaction>
            <BusinessTransaction name="Inventory Report">
                  <RequestingBusinessActivity name="">
                        <DocumentEnvelope isPositiveResponse="true"
businessDocument="Inventory Report"/>
                  </RequestingBusinessActivity>
                  <RespondingBusinessActivity name=""/>
            </BusinessTransaction>
      </Package>
</ProcessSpecification>
```

Appendix B: Business Process Specification Schema DTD

```
<!-- ============================================================= --
>
<!--  Editor:   Kurt Kanaskie (Lucent Technologies)                --
>
<!--  Version:  Version 1.0                                        --
>
<!--  Updated:  2001-04-27                                         --
>
<!--                                                               --
>
<!--  Public Identifier:                                           --
>
<!--            "-//ebXML//DTD Process Specification ver 1.0//EN"  --
>
<!--                                                               --
>
<!--  Purpose:                                                     --
>
<!--     The ebXML Specification DTD provides a standard           --
>
<!--     framework by which business systems may be                --
>
<!--     configured to support execution of business               --
>
<!--     transactions.  It is based upon prior UN/CEFACT           --
>
<!--     work, specifically the metamodel behind the               --
>
<!--     UN/CEFACT Unified Modeling Methodology (UMM) defined      --
>
<!--     in the N090R9.1  specification.
-->
<!--                                                               --
>
<!--     The Specification Schema supports the specification       --
>
<!--     of Business Transactions and the choreography of          --
>
<!--     Business Transactions into Business Collaborations.       --
>
<!--                                                               --
>
<!--  Notes:                                                       --
>
<!--     time periods are represented using ISO 8601 format        --
>
<!--     (e.g. P2D for 2 Days, P2H30M for 2 Hours 30 Minutes       --
>
```

```
<!--                                                                --
>
<!--     Naming and reference is based on convention that an         --
>
<!--     Element with a name attribute (e.g. AuthorizedRole)        --
>
<!--     is refernced by an attribute in another element with       --
>
<!--     the name in lowerCamelCase (e.g. authorizedRole).          --
>
<!--                                                                --
>
<!--     fromBusinessState and toBusinessState refer to the         --
>
<!--     the names of a BusinessTransactionActivity,                --
>
<!--     CollaborationActivity, Fork, and Join, all are targets for --
>
<!--     from/to in Transition. This deviates from the normal       --
>
<!--     convention of lowerCamelCase attribute name                --
>
<!--     BusinessState is used as a generic term for:               --
>
<!--     Fork, Join, Success, Failure                               --
>
<!--                                                                --
>
<!-- Constraints:                                                   --
>
<!--     - attributes location, logicalModel, pattern, specification --
>
<!--     uri, are of type xsd:anyURI                                -->
<!--     - attributes timeTo* are of type xsd:duration               --
>
<!--       isSuccess is an expression (e.g. XPath) that results      --
>
<!--       in a boolean true or false based on document name or      --
>
<!--       document content.                                         --
>
<!--                                                                --
>
<!-- ============================================================== --
>

<!ELEMENT ProcessSpecification (Documentation*, (Include* |
DocumentSpecification* | ProcessSpecification* |
                          Package | BinaryCollaboration |
BusinessTransaction | MultiPartyCollaboration)*)>
<!ATTLIST ProcessSpecification
      name ID #REQUIRED
      uuid CDATA #REQUIRED
      version CDATA #REQUIRED
>
```

```
<!ELEMENT Documentation (#PCDATA)>
<!ATTLIST Documentation
      uri CDATA #IMPLIED
>

<!ELEMENT Include (Documentation*)>
<!ATTLIST Include
      name CDATA #REQUIRED
      uuid CDATA #REQUIRED
      uri CDATA #REQUIRED
      version CDATA #REQUIRED
>
<!ELEMENT DocumentSpecification (Documentation*, BusinessDocument*)>
<!ATTLIST DocumentSpecification
      name CDATA #REQUIRED
      nameID ID #IMPLIED
      location CDATA #IMPLIED
      logicalModel CDATA #IMPLIED
      version CDATA #IMPLIED
>
<!ELEMENT BusinessDocument (Documentation*)>
<!ATTLIST BusinessDocument
      name CDATA #REQUIRED
      nameID ID #IMPLIED
>
<!ELEMENT Package (Documentation*,
               (Package | BinaryCollaboration | BusinessTransaction |
MultiPartyCollaboration)*)>
<!ATTLIST Package
      name CDATA #REQUIRED
      nameID ID #IMPLIED
>
<!-- Model requires 2 Authorized roles -->
<!ELEMENT BinaryCollaboration (Documentation*, AuthorizedRole,
AuthorizedRole,
                        (Documentation* | Start | Transition | Success
| Failure |
                         BusinessTransactionActivity |
CollaborationActivity | Fork | Join)*)>
<!ATTLIST BinaryCollaboration
      name CDATA #REQUIRED
      nameID ID #IMPLIED
      pattern CDATA #IMPLIED
      beginsWhen CDATA #IMPLIED
      endsWhen CDATA #IMPLIED
      preCondition CDATA #IMPLIED
      postCondition CDATA #IMPLIED
      timeToPerform CDATA #IMPLIED
>
<!ELEMENT MultiPartyCollaboration (Documentation*,
BusinessPartnerRole*)>
<!ATTLIST MultiPartyCollaboration
      name CDATA #REQUIRED
      nameID ID #IMPLIED
>
```

```
<!ELEMENT AuthorizedRole (Documentation*)>
<!ATTLIST AuthorizedRole
      name CDATA #REQUIRED
      nameID ID #IMPLIED
>

<!-- A BusinessState is one of Start, Success, Failure, Fork, Join,
BusinessTransactionActivity or CollaborationActivity -->
<!-- fromBusinessState and toBusinessState are fully qualified using
XPath -->
<!-- "guardExpression" is an expression that results in a boolean true
or false -->
<!ELEMENT Transition (Documentation*)>
<!ATTLIST Transition
      onInitiation (true | false) "false"
      fromBusinessState CDATA #IMPLIED
      fromBusinessStateIDRef IDREF #IMPLIED
      toBusinessState CDATA #IMPLIED
      toBusinessStateIDRef IDREF #IMPLIED
      guardCondition (Success | BusinessFailure | TechnicalFailure |
AnyFailure ) #IMPLIED
      guardExpression CDATA #IMPLIED
>
<!-- Start is a special type of Transition in that it only has a
destination -->
<!ELEMENT Start (Documentation*)>
<!ATTLIST Start
      toBusinessState CDATA #REQUIRED
      toBusinessStateIDRef IDREF #IMPLIED
>
<!-- Success is a special type of Transition in that it only has a
origination -->
<!ELEMENT Success (Documentation*)>
<!ATTLIST Success
      fromBusinessState CDATA #REQUIRED
      fromBusinessStateIDRef IDREF #IMPLIED
      guardCondition (Success | BusinessFailure | TechnicalFailure |
AnyFailure ) #IMPLIED
      guardExpression CDATA #IMPLIED
>
<!-- Failure is a special type of Transition in that it only has a
origination -->
<!-- guardExpression is an expression (e.g. XPath on BusinessDocument)
that results in a boolean result -->
<!ELEMENT Failure (Documentation*)>
<!ATTLIST Failure
      fromBusinessState CDATA #REQUIRED
      fromBusinessStateIDRef IDREF #IMPLIED
      guardCondition (Success | BusinessFailure | TechnicalFailure |
AnyFailure ) #IMPLIED
      guardExpression CDATA #IMPLIED
>
```

```
<!-- Fork is a special type of BusinessState that can be transitioned
to -->
<!ELEMENT Fork (Documentation*)>
<!ATTLIST Fork
      name CDATA #REQUIRED
      nameID ID #IMPLIED
>
<!-- Fork is a special type of BusinessState that can be transitioned
to -->
<!ELEMENT Join (Documentation*)>
<!ATTLIST Join
      name CDATA #REQUIRED
      nameID ID #IMPLIED
      waitForAll (true | false) "true"
>

<!-- fromAuthorizedRole and toAuthorizedRole are fully qualified using
XPath -->
<!-- BusinessTransactionActivity is a BusinessState that can be
transitioned to -->
<!ELEMENT BusinessTransactionActivity (Documentation*)>
<!ATTLIST BusinessTransactionActivity
      name CDATA #REQUIRED
      nameID ID #IMPLIED
      businessTransaction CDATA #REQUIRED
      businessTransactionIDRef IDREF #IMPLIED
      fromAuthorizedRole CDATA #REQUIRED
      fromAuthorizedRoleIDRef IDREF #IMPLIED
      toAuthorizedRole CDATA #REQUIRED
      toAuthorizedRoleIDRef IDREF #IMPLIED
      isConcurrent (true | false) "true"
      isLegallyBinding (true | false) "true"
      timeToPerform CDATA #IMPLIED
>

<!-- fromAuthorizedRole and toAuthorizedRole are fully qualified using
XPath -->
<!-- CollaborationActivity is a BusinessState that can be transitioned
to -->
<!ELEMENT CollaborationActivity (Documentation*)>
<!ATTLIST CollaborationActivity
      name CDATA #REQUIRED
      nameID ID #IMPLIED
      fromAuthorizedRole CDATA #REQUIRED
      fromAuthorizedRoleIDRef IDREF #IMPLIED
      toAuthorizedRole CDATA #REQUIRED
      toAuthorizedRoleIDRef IDREF #IMPLIED
      binaryCollaboration CDATA #REQUIRED
      binaryCollaborationIDRef IDREF #IMPLIED
>
<!ELEMENT BusinessTransaction (Documentation*,
RequestingBusinessActivity, RespondingBusinessActivity)>
<!ATTLIST BusinessTransaction
      name CDATA #REQUIRED
      nameID ID #IMPLIED
```

```
      pattern CDATA #IMPLIED
      beginsWhen CDATA #IMPLIED
      endsWhen CDATA #IMPLIED
      isGuaranteedDeliveryRequired (true | false) "false"
      preCondition CDATA #IMPLIED
      postCondition CDATA #IMPLIED
>
<!ELEMENT RequestingBusinessActivity (Documentation*,
DocumentEnvelope)>
<!ATTLIST RequestingBusinessActivity
      name CDATA #IMPLIED
      nameID ID #IMPLIED
      isAuthorizationRequired (true | false) "false"
      isIntelligibleCheckRequired (true | false) "false"
      isNonRepudiationReceiptRequired (true | false) "false"
      isNonRepudiationRequired (true | false) "false"
      timeToAcknowledgeAcceptance CDATA #IMPLIED
      timeToAcknowledgeReceipt CDATA #IMPLIED
>
<!ELEMENT RespondingBusinessActivity (Documentation*,
DocumentEnvelope*)>
<!ATTLIST RespondingBusinessActivity
      name CDATA #IMPLIED
      nameID ID #IMPLIED
      isAuthorizationRequired (true | false) "false"
      isIntelligibleCheckRequired (true | false) "false"
      isNonRepudiationReceiptRequired (true | false) "false"
      isNonRepudiationRequired (true | false) "false"
      timeToAcknowledgeReceipt CDATA #IMPLIED
>
<!-- isPositiveResponse is an expression (e.g. XPath on
BusinessDocument) that results in a boolean result -->
<!ELEMENT DocumentEnvelope (Documentation*, Attachment*)>
<!ATTLIST DocumentEnvelope
      businessDocument CDATA #REQUIRED
      businessDocumentIDRef IDREF #IMPLIED
      isPositiveResponse CDATA #IMPLIED
      isAuthenticated (true | false) "false"
      isConfidential (true | false) "false"
      isTamperProof (true | false) "false"
>
<!ELEMENT Attachment (Documentation*)>
<!ATTLIST Attachment
      name CDATA #REQUIRED
      nameID ID #IMPLIED
      businessDocument CDATA #IMPLIED
      businessDocumentIDRef IDREF #IMPLIED
      mimeType CDATA #REQUIRED
      specification CDATA #IMPLIED
      version CDATA #IMPLIED
      isAuthenticated (true | false) "false"
      isConfidential (true | false) "false"
      isTamperProof (true | false) "false"
>
<!ELEMENT BusinessPartnerRole (Documentation*, Performs*, Transition*)>
```

```
<!ATTLIST BusinessPartnerRole
      name CDATA #REQUIRED
      nameID ID #IMPLIED
>

<!-- authorizedRole is fully qualified using XPath -->
<!ELEMENT Performs (Documentation*)>
<!ATTLIST Performs
      authorizedRole CDATA #REQUIRED
      authorizedRoleIDRef IDREF #IMPLIED
>
```

Appendix C: Business Process Specification Schema XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Kurt
Kanaskie (Lucent Technologies) -->
<!--W3C Schema generated by XML Spy v3.5 NT (http://www.xmlspy.com)-->
<!-- Kanaskie Updated 2001-04-27
      This is the version that works with XML Spy 3.5
            Use uriReference instead of anyURI
            Use timeDuration instead of duration
-->
<!-- Kanaskie Changed 2001-04-27
      See DTD for list of changes.
      Differences from DTD version:
      AuthorizedRole minOccurs=2 maxOccurs=2
      <xsd:attribute name="pattern" type="xsd:anyURI"/>
      <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="location" type="xsd:anyURI"/>
      <xsd:attribute name="logicalModel" type="xsd:anyURI"/>
      <xsd:attribute name="specification" type="xsd:anyURI"/>
      <xsd:attribute name="timeToPerform" type="xsd:duration"/>
      <xsd:attribute name="timeToPerform" type="xsd:duration"/>
      <xsd:attribute name="timeToAcknowledgeAcceptance"
type="xsd:duration"/>
      <xsd:attribute name="timeToAcknowledgeReceipt"
type="xsd:duration"/>
      <xsd:attribute name="timeToAcknowledgeAcceptance"
type="xsd:duration"/>
      <xsd:attribute name="timeToAcknowledgeReceipt"
type="xsd:duration"/>
      <xsd:attribute name="isAuthenticated" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="isConfidential" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="isTamperProof" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="isGuaranteedDeliveryRequired"
type="xsd:boolean" value="false"/>
      <xsd:attribute name="isConcurrent" type="xsd:boolean"
value="true"/>
      <xsd:attribute name="isLegallyBinding" type="xsd:boolean"
value="true"/>
      <xsd:attribute name="isAuthenticated" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="isConfidential" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="isTamperProof" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="waitForAll" type="xsd:boolean"
value="true"/>
      <xsd:attribute name="isAuthorizationRequired" type="xsd:boolean"
value="false"/>
```

```
      <xsd:attribute name="isIntelligibleCheckRequired"
type="xsd:boolean" value="false"/>
      <xsd:attribute name="isNonRepudiationReceiptRequired"
type="xsd:boolean" value="false"/>
      <xsd:attribute name="isNonRepudiationRequired" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="isAuthorizationRequired" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="isIntelligibleCheckRequired"
type="xsd:boolean" value="false"/>
      <xsd:attribute name="isNonRepudiationReceiptRequired"
type="xsd:boolean" value="false"/>
      <xsd:attribute name="isNonRepudiationRequired" type="xsd:boolean"
value="false"/>
      <xsd:attribute name="onInitiation" type="xsd:boolean"
value="false"/>
-->
<xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns="http://www.ebxml.org/BusinessProcess"
elementFormDefault="qualified">
      <xsd:element name="Attachment">
            <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="nameID" type="xsd:ID"/>
                  <xsd:attribute name="businessDocument"
type="xsd:string"/>
                  <xsd:attribute name="businessDocumentIDRef"
type="xsd:IDREF"/>
                  <xsd:attribute name="specification"
type="xsd:uriReference"/>
                  <xsd:attribute name="mimeType" type="xsd:string"
use="required"/>
                  <xsd:attribute name="version" type="xsd:string"/>
                  <xsd:attribute name="isAuthenticated"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isConfidential"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isTamperProof"
type="xsd:boolean" value="false"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="AuthorizedRole">
            <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="nameID" type="xsd:ID"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="BinaryCollaboration">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element ref="AuthorizedRole"/>
                        <xsd:element ref="AuthorizedRole"/>
                        <xsd:choice minOccurs="0"
maxOccurs="unbounded">
                              <xsd:element ref="Documentation"
minOccurs="0" maxOccurs="unbounded"/>
                              <xsd:element ref="Start"/>
                              <xsd:element ref="Transition"/>
                              <xsd:element ref="Success"/>
                              <xsd:element ref="Failure"/>
                              <xsd:element
ref="BusinessTransactionActivity"/>
                              <xsd:element
ref="CollaborationActivity"/>
                              <xsd:element ref="Fork"/>
                              <xsd:element ref="Join"/>
                        </xsd:choice>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string"
use="required"/>
                  <xsd:attribute name="nameID" type="xsd:ID"/>
                  <xsd:attribute name="pattern"
type="xsd:uriReference"/>
                  <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:timeDuration"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="BusinessDocument">
            <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="nameID" type="xsd:ID"/>
            </xsd:complexType>
      </xsd:element>
      <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:attribute name="name" type="xsd:string"
use="required"/>
                  <xsd:attribute name="nameID" type="xsd:ID"/>
            </xsd:complexType>
      </xsd:element>
      <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:attribute name="name" type="xsd:string"
use="required"/>
                  <xsd:attribute name="nameID" type="xsd:ID"/>
                  <xsd:attribute name="pattern"
type="xsd:uriReference"/>
                  <xsd:attribute name="beginsWhen" type="xsd:string"/>
                  <xsd:attribute name="endsWhen" type="xsd:string"/>
                  <xsd:attribute name="isGuaranteedDeliveryRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="preCondition"
type="xsd:string"/>
                  <xsd:attribute name="postCondition"
type="xsd:string"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="BusinessTransactionActivity">
            <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="nameID" type="xsd:ID"/>
                  <xsd:attribute name="businessTransaction"
type="xsd:string" use="required"/>
                  <xsd:attribute name="businessTransactionIDRef"
type="xsd:IDREF"/>
                  <xsd:attribute name="fromAuthorizedRole"
type="xsd:string" use="required"/>
                  <xsd:attribute name="fromAuthorizedRoleIDRef"
type="xsd:IDREF"/>
                  <xsd:attribute name="toAuthorizedRole"
type="xsd:string" use="required"/>
```

```
4101                    <xsd:attribute name="toAuthorizedRoleIDRef"
4102  type="xsd:IDREF"/>
4103                    <xsd:attribute name="isConcurrent" type="xsd:boolean"
4104  value="true"/>
4105                    <xsd:attribute name="isLegallyBinding"
4106  type="xsd:boolean" value="true"/>
4107                    <xsd:attribute name="timeToPerform"
4108  type="xsd:timeDuration"/>
4109              </xsd:complexType>
4110        </xsd:element>
4111        <xsd:element name="CollaborationActivity">
4112              <xsd:complexType>
4113                    <xsd:sequence>
4114                          <xsd:element ref="Documentation" minOccurs="0"
4115  maxOccurs="unbounded"/>
4116                    </xsd:sequence>
4117                    <xsd:attribute name="nameID" type="xsd:ID"/>
4118                    <xsd:attribute name="name" type="xsd:string"
4119  use="required"/>
4120                    <xsd:attribute name="fromAuthorizedRole"
4121  type="xsd:string" use="required"/>
4122                    <xsd:attribute name="fromAuthorizedRoleIDRef"
4123  type="xsd:IDREF"/>
4124                    <xsd:attribute name="toAuthorizedRole"
4125  type="xsd:string" use="required"/>
4126                    <xsd:attribute name="toAuthorizedRoleIDRef"
4127  type="xsd:IDREF"/>
4128                    <xsd:attribute name="binaryCollaboration"
4129  type="xsd:string" use="required"/>
4130                    <xsd:attribute name="binaryCollaborationIDRef"
4131  type="xsd:IDREF"/>
4132              </xsd:complexType>
4133        </xsd:element>
4134        <xsd:element name="DocumentEnvelope">
4135              <xsd:complexType>
4136                    <xsd:sequence>
4137                          <xsd:element ref="Documentation" minOccurs="0"
4138  maxOccurs="unbounded"/>
4139                          <xsd:element ref="Attachment" minOccurs="0"
4140  maxOccurs="unbounded"/>
4141                    </xsd:sequence>
4142                    <xsd:attribute name="businessDocument"
4143  type="xsd:string" use="required"/>
4144                    <xsd:attribute name="businessDocumentIDRef"
4145  type="xsd:IDREF"/>
4146                    <xsd:attribute name="isPositiveResponse"
4147  type="xsd:string"/>
4148                    <xsd:attribute name="isAuthenticated"
4149  type="xsd:boolean" value="false"/>
4150                    <xsd:attribute name="isConfidential"
4151  type="xsd:boolean" value="false"/>
4152                    <xsd:attribute name="isTamperProof"
4153  type="xsd:boolean" value="false"/>
4154              </xsd:complexType>
4155        </xsd:element>
```

```
4156        <xsd:element name="DocumentSpecification">
4157              <xsd:complexType>
4158                    <xsd:sequence>
4159                          <xsd:element ref="Documentation" minOccurs="0"
4160  maxOccurs="unbounded"/>
4161                          <xsd:element ref="BusinessDocument"
4162  minOccurs="0" maxOccurs="unbounded"/>
4163                    </xsd:sequence>
4164                    <xsd:attribute name="name" type="xsd:string"
4165  use="required"/>
4166                    <xsd:attribute name="nameID" type="xsd:ID"/>
4167                    <xsd:attribute name="location"
4168  type="xsd:uriReference"/>
4169                    <xsd:attribute name="logicalModel"
4170  type="xsd:uriReference"/>
4171                    <xsd:attribute name="version" type="xsd:string"/>
4172              </xsd:complexType>
4173        </xsd:element>
4174        <xsd:element name="Documentation">
4175              <xsd:complexType>
4176                    <xsd:simpleContent>
4177                          <xsd:restriction base="xsd:string">
4178                                <xsd:attribute name="uri"
4179  type="xsd:uriReference"/>
4180                          </xsd:restriction>
4181                    </xsd:simpleContent>
4182              </xsd:complexType>
4183        </xsd:element>
4184        <xsd:element name="Failure">
4185              <xsd:complexType>
4186                    <xsd:sequence>
4187                          <xsd:element ref="Documentation" minOccurs="0"
4188  maxOccurs="unbounded"/>
4189                    </xsd:sequence>
4190                    <xsd:attribute name="fromBusinessState"
4191  type="xsd:string" use="required"/>
4192                    <xsd:attribute name="fromBusinessStateIDRef"
4193  type="xsd:IDREF"/>
4194                    <xsd:attribute name="guardCondition">
4195                          <xsd:simpleType>
4196                                <xsd:restriction base="xsd:NMTOKEN">
4197                                      <xsd:enumeration value="Success"/>
4198                                      <xsd:enumeration
4199  value="BusinessFailure"/>
4200                                      <xsd:enumeration
4201  value="TechnicalFailure"/>
4202                                      <xsd:enumeration
4203  value="AnyFailure"/>
4204                                </xsd:restriction>
4205                          </xsd:simpleType>
4206                    </xsd:attribute>
4207                    <xsd:attribute name="guardExpression"
4208  type="xsd:string"/>
4209              </xsd:complexType>
4210        </xsd:element>
```

```
      <xsd:element name="Fork">
            <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="nameID" type="xsd:ID"/>
            </xsd:complexType>
      </xsd:element>
      <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:string"
use="required"/>
                  <xsd:attribute name="uri" type="xsd:uriReference"
use="required"/>
                  <xsd:attribute name="version" type="xsd:string"
use="required"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="Join">
            <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="nameID" type="xsd:ID"/>
                  <xsd:attribute name="waitForAll" type="xsd:boolean"
value="true"/>
            </xsd:complexType>
      </xsd:element>
      <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:attribute name="name" type="xsd:string"
use="required"/>
                  <xsd:attribute name="nameID" type="xsd:ID"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="Package">
            <xsd:complexType>
```

```
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:choice minOccurs="0"
maxOccurs="unbounded">
                              <xsd:element ref="Package"/>
                              <xsd:element ref="BinaryCollaboration"/>
                              <xsd:element ref="BusinessTransaction"/>
                              <xsd:element
ref="MultiPartyCollaboration"/>
                        </xsd:choice>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string"
use="required"/>
                  <xsd:attribute name="nameID" type="xsd:ID"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="Performs">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="authorizedRole"
type="xsd:string" use="required"/>
                  <xsd:attribute name="authorizedRoleIDRef"
type="xsd:IDREF"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="ProcessSpecification">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:choice minOccurs="0"
maxOccurs="unbounded">
                              <xsd:element ref="Include" minOccurs="0"
maxOccurs="unbounded"/>
                              <xsd:element ref="DocumentSpecification"
minOccurs="0" maxOccurs="unbounded"/>
                              <xsd:element ref="ProcessSpecification"
minOccurs="0" maxOccurs="unbounded"/>
                              <xsd:element ref="Package"/>
                              <xsd:element ref="BinaryCollaboration"/>
                              <xsd:element ref="BusinessTransaction"/>
                              <xsd:element
ref="MultiPartyCollaboration"/>
                        </xsd:choice>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:ID"
use="required"/>
                  <xsd:attribute name="uuid" type="xsd:string"
use="required"/>
                  <xsd:attribute name="version" type="xsd:string"
use="required"/>
```

```
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="RequestingBusinessActivity">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element ref="DocumentEnvelope"/>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string"
use="required"/>
                  <xsd:attribute name="nameID" type="xsd:ID"/>
                  <xsd:attribute name="isAuthorizationRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isIntelligibleCheckRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isNonRepudiationReceiptRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isNonRepudiationRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="timeToAcknowledgeAcceptance"
type="xsd:timeDuration"/>
                  <xsd:attribute name="timeToAcknowledgeReceipt"
type="xsd:timeDuration"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="RespondingBusinessActivity">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element ref="DocumentEnvelope"
minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string"
use="required"/>
                  <xsd:attribute name="nameID" type="xsd:ID"/>
                  <xsd:attribute name="isAuthorizationRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isIntelligibleCheckRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isNonRepudiationReceiptRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="isNonRepudiationRequired"
type="xsd:boolean" value="false"/>
                  <xsd:attribute name="timeToAcknowledgeReceipt"
type="xsd:timeDuration"/>
            </xsd:complexType>
      </xsd:element>
      <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>
      <xsd:element name="Success">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="fromBusinessState"
type="xsd:string" use="required"/>
                  <xsd:attribute name="fromBusinessStateIDRef"
type="xsd:IDREF"/>
                  <xsd:attribute name="guardCondition">
                        <xsd:simpleType>
                              <xsd:restriction base="xsd:NMTOKEN">
                                    <xsd:enumeration value="Success"/>
                                    <xsd:enumeration
value="BusinessFailure"/>
                                    <xsd:enumeration
value="TechnicalFailure"/>
                                    <xsd:enumeration
value="AnyFailure"/>
                              </xsd:restriction>
                        </xsd:simpleType>
                  </xsd:attribute>
                  <xsd:attribute name="guardExpression"
type="xsd:string"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="Transition">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="Documentation" minOccurs="0"
maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="onInitiation" type="xsd:boolean"
value="false"/>
                  <xsd:attribute name="fromBusinessState"
type="xsd:string"/>
                  <xsd:attribute name="fromBusinessStateIDRef"
type="xsd:IDREF"/>
                  <xsd:attribute name="toBusinessState"
type="xsd:string"/>
                  <xsd:attribute name="toBusinessStateIDRef"
type="xsd:IDREF"/>
                  <xsd:attribute name="guardCondition">
                        <xsd:simpleType>
                              <xsd:restriction base="xsd:NMTOKEN">
                                    <xsd:enumeration value="Success"/>
                                    <xsd:enumeration
value="BusinessFailure"/>
```

```
                                        <xsd:enumeration
value="TechnicalFailure"/>
                                        <xsd:enumeration
value="AnyFailure"/>
                                </xsd:restriction>
                           </xsd:simpleType>
                    </xsd:attribute>
                    <xsd:attribute name="guardExpression"
type="xsd:string"/>
              </xsd:complexType>
        </xsd:element>
</xsd:schema>
```

11 References

1. UN/CEFACT Modelling Methodology (CEFACT/TMWG/N090R9.1)
2. RosettaNet Implementation Framework: Core Specification, Version: Release 2.00.00, 3 January 2001

12 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.

13 Contact Information

Team Leader (Of the BP team):
Paul Levine
Telcordia Technologies, Inc.
45 Knightsbridge Road
Piscataway, N.J. 08854
US

Phone: 732-699-3042
EMail: plevine@telcordia.com

Sub Team Lead (Of the context/MetamodelGroup) :

Karsten Riemer
Sun Microsystems
1 Network Drive
Burlington, MA 01803
USA

Phone: 781-442-2679
EMail: karsten.riemer@sun.com

Editor (of this document):

Karsten Riemer
Sun Microsystems
1 Network Drive
Burlington, MA 01803
USA

Phone: 781-442-2679
EMail: karsten.riemer@sun.com