Creating A Single Global Electronic Market

ebXML Business Process Specification Schema Version 0.99

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

03/19/2001

1 Status of this Document

This document is a working 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_0.99

***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, I.C.O.T. - 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)
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
Antonio Carrasco, Data Access Technologies
Neal Smith, Chevron

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
Executive Summary v
4.1 Summary of Contents of Document 1
4.2 Audience 1
4.3 Related Documents 1
4.4 Prerequisites 1
5 Design Objectives 2
5.1 Goals/Objectives/Requirements/Problem Description 2
5.2 Caveats and Assumptions 2
5.2.1 Relationship between Specification Schema and UMM 2
6 System Overview 4
UML Specification Schema 6
DTD Specification Schema 6
Business Process Interaction Patterns 6
Common Modeling Elements 6
Production Rules 7
6.1 Key Concepts of the Specification Schema 7
6.2 How to use the ebXML Business Process Specification Schema 10
6.3 How Specification Schema is used with other ebXML specifications 11
6.4 How to design collaborations and transactions, re-using at design time 13
6.4.1 Specify a Business Transaction and its Document Flow 14
6.4.2 Specify a Binary Collaboration 20
6.4.3 Specify a MultiParty Collaboration 23
6.4.4 Specify a Choreography 25
6.4.5 The whole model 27
6.5 Core Business Transaction Semantics 29
6.5.1 Defining and using transaction and interaction patterns 30
6.5.2 Transaction Patterns 30
6.5.3 Parameters required for CPP/CPA 30
6.5.4 Delivery Channel selection parameters: 31
6.5.5 Reliability 32
6.5.6 Synchronous or Asynchronous 32
6.5.7 BSI service level parameters: 33
6.6 Run time transaction semantics 34
6.6.1 Timeouts 35
6.6.2 Exceptions 36
6.6.3 ControlException 36
6.6.4 Business Protocol Exceptions (a.k.a. ProcessException) 37
6.7 Runtime Collaboration Semantics 38
6.8 Where the ebXML Specification Schema May Be Implemented 38

7 UML Element Specification 38
7.1 Business Collaborations 38
7.1.1 MultiPartyCollaboration 39
7.1.2 BusinessPartnerRole 39
7.1.3 Performs 39
7.1.4 AuthorizedRole 40
7.1.5 BinaryCollaboration 41
7.1.6 BusinessActivity 42
7.1.7 BusinessTransactionActivity 42
7.1.8 CollaborationActivity 43
7.2 Business Transactions 43
7.2.1 BusinessTransaction 43
7.2.2 Business Action 45
7.2.3 RequestingBusinessActivity 46
7.2.4 RespondingBusinessActivity 46
7.3 Document Flow 47
7.3.1 Document Flow 47
7.3.2 DocumentType 48
7.3.3 Schema 48
7.3.4 Attachment 49
7.4 Choreography within Collaborations. 49
7.4.1 BusinessState 49
7.4.2 Transition 50
7.4.3 Start 50
7.4.4 TerminalState 51
7.4.5 Success 51
7.4.6 Failure 51
7.4.7 Fork 52
7.4.8 Join 52
7.4.9 Guard 52
7.5 Definition and Scope 53
7.6 Collaboration Specification Rules 53
7.6.1 Well-formedness Rules 53
8 Specification Schema – (DTD) 55
8.1 DTD 55
8.2 Documentation for the DTD 59
8.3 XML to UML cross-reference 82
8.4 Scoped Name Reference 84
8.5 Sample XML document against above DTD 85
9 Common Modeling Elements 91
9.1 Data typing 91
9.1.1 Global Data types 91
9.1.2 Local Datatypes 94
9.2 Business signal structures 94
9.2.1 ReceiptAcknowledgment DTD 94

9.2.2 AcceptanceAcknowledgement DTD 94
9.2.3 Exception Signal DTD 95
10 Production Rules 96
11 References 98
12 Disclaimer 98
13 Contact Information 99
Copyright Statement 100

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 Unified Modeling Methodology (UMM) defined in the N90 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 messages 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 Collaboration Protocol Profile and Agreement, version 1.0
- ebXML Business Process Analysis Worksheets

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 specify 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.

The *Specification Schema* is available in two stand-alone representations, a UML profile, and a DTD. Users of the Specification Schema will create business process specifications as either UML diagrams, or Extensible Markup Language (XML) documents.

The objective of the UML based *Specification Schema* is to provide sufficient functionality for business people to model a collaboration. The objective of the XML based *Specification Schema* is to provide sufficient functionality for technical people to unambiguously derive from that collaboration model a set of runtime configuration parameters. The UML and XML based versions of the *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 Unified Modeling Methodology (UMM).

The 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 Business Document model included in the ebXML Core Components specifications.

5.2.1 Relationship between Specification Schema and UMM

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

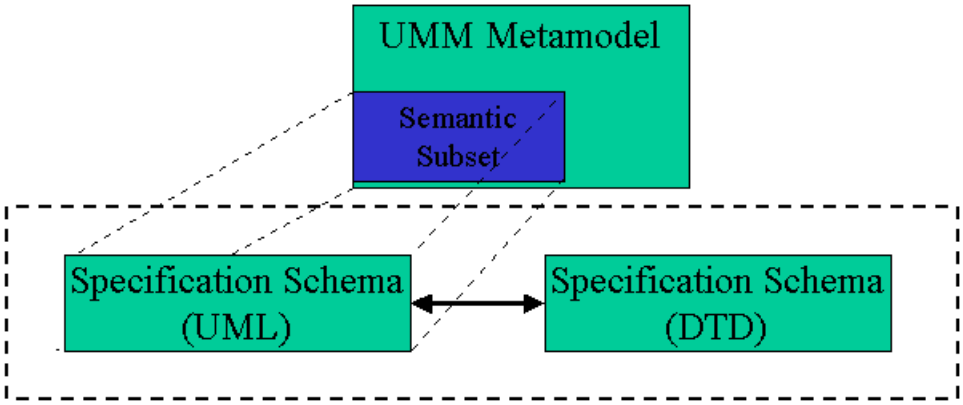
interaction between roles takes place as a choreographed set of Business Transactions. Each Business Transaction is expressed as an exchange of electronic Business Documents. The sequence of the exchange is determined by the Business Process, and by messaging and security considerations. Business Documents are composed from re-useable business information components. At a lower level, Business Processes can be composed of re-useable Core Processes, and Business Objects can be composed of re-useable Core Components.

The UMM Meta Model supports requirements, analysis and design 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.

An additional view of the metamodel, the Specification Schema, is also 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 Specification Schema forms a semantic subset of the UMM Meta Model. The Specification Schema is available in two stand-alone representations, a UML profile, and a DTD.

The relationship between the UMM Meta Model and the ebXML Specification Schema can be shown as follows:

Figure 1. Relationship between Metamodel and Specification Schema



6 System Overview

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 messages and business signals between the partners to achieve the required electronic commerce transaction.

To help specify the patterns *the Specification Schema* is accompanied by a set of modeling elements common to those standard patterns. The full specification, thus, of a business process consists of a business process model specified against the *Specification Schema* and an identification of the desired pattern(s). This full specification is then the input to the formation of trading partner Collaboration Protocol Profiles and Collaboration Protocol Agreements.

This can be shown as follows:

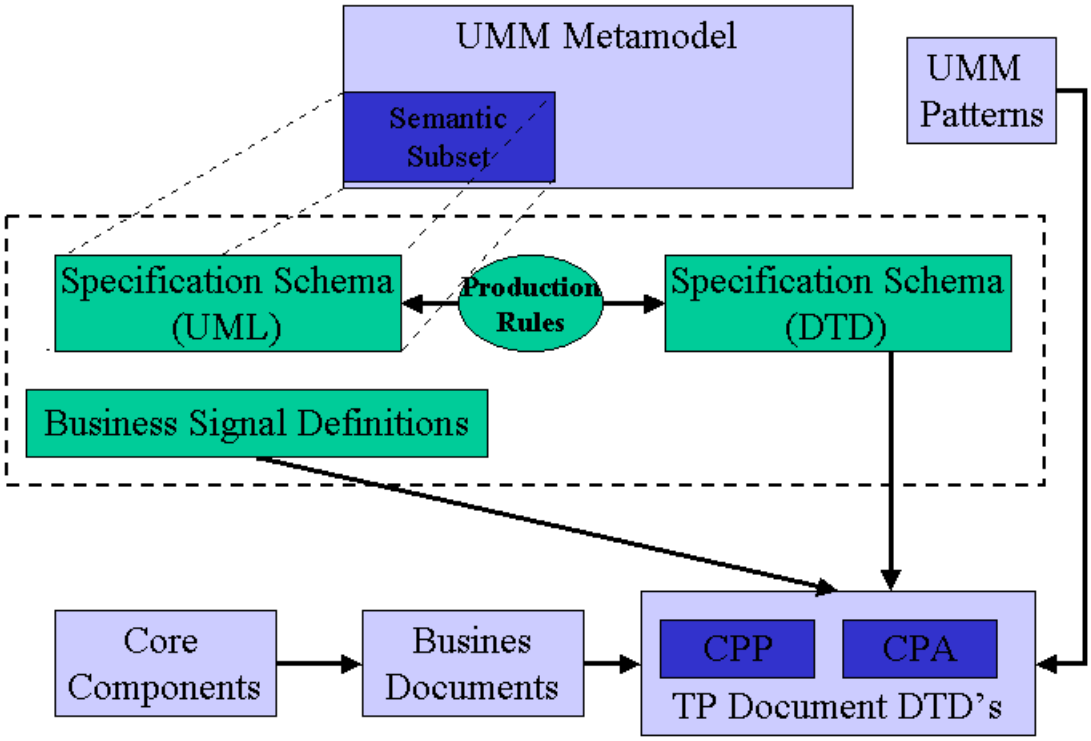


Figure 2. Relationship of specification schema to TP and Core Components

As the figure shows, the architecture of the ebXML Specification Schema consists of the following functional components (shown inside the dotted box):

- UML Specification Schema
- DTD specification Schema
- Business Signal Definitions
- Production Rules needed for the generation of UML specification into a XML Specification Document

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

In addition, the UMM provides a set of Business Transaction Interaction Patterns

The Specification Schema does not by itself define Business Documents. Rather it points to already existing Business Document definitions. Such definitions may have been defined based on ebXML core components, or may be supplied from some other source.

This run time business process and information specification is then incorporated with the associated Business Document definitions into trading partner

Collaboration Protocol Profiles (CPP) and Collaboration Protocol Agreements (CPA). Within these profiles and agreements are then added further technical parameters resulting in a full specification of the run-time software at each trading partner.

Each of the components in the Specification Schema is described below:

UML Specification Schema

The UML Specification Schema is a semantic subset of the metamodel behind UMM as specified in UN/CEFACT TMWG's N90, expressed as a standalone UML profile. The UML Specification Schema will through the application of production rules produce an XML Specification Document 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.

DTD Specification Schema

The DTD Specification Schema is an isomorphic definition of the UML Specification Schema. The DTD Specification Schema seeks to guarantee that a XML Specification Document is analytically, semantically and functionally equivalent to a UML Specification Model of the same business process.

This version of the specification is expressed as a DTD, and some of the constraints may need to be stated separately, in plain text. It is the intent to migrate to W3C schema, as soon as it becomes available as a standard. At that point such constraints, where possible, will be built into the schema.

Business Process Interaction Patterns

ebXML business service interfaces are configured to execute the business processes defined against the specification schema. They do so by exchanging ebXML messages and business signals. The Business Process Interaction Patterns set forth in Chapter [11] of the UMM N90 document illustrate the permissible permutations of message sequences as determined by the type of business transaction defined and the timing policies specified in the transactions. Those timing policies are expressed by use of the security and timing parameters provided in this document.

Common Modeling Elements

The Common Modeling Elements specifies the modeling elements, and their interrelationships, that are used to specify and use Interaction Patterns.

Business signals are such common modeling elements.

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, business signals are ‘constant’ and do not vary from transaction to transaction. Thus, they can be defined once and for all as part of the common model elements.

Production Rules

The Specification Production rules provide the prescriptive definition necessary to translate a UML Specification Model into an XML Specification Document and the well-formed rules necessary to populate an XML Specification Document.

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

6.1 Key Concepts of the Specification Schema

The ebXML 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 can be shown as follows:

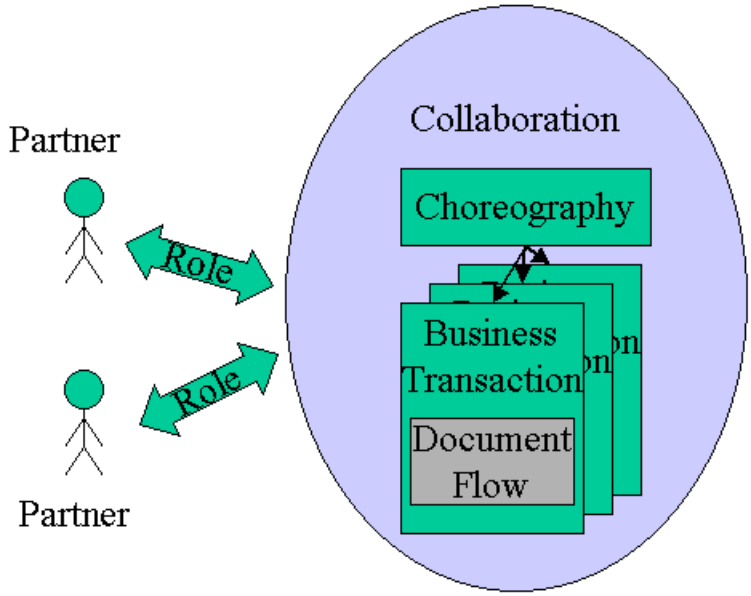


Figure 3. Basic Semantics of a business collaboration

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.

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 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 BusinessActivities between the two roles. Each BusinessActivity reflects a state in the collaboration. The BusinessActivity can be ‘atomic’, i.e. the activity of conducting an atomic BusinessTransaction, or ‘composite’, i.e. the activity of conducting another Binary Collaboration. In either case the activities can be choreographed 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. In essence 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. a Business Service Interface (BSI).

A Business Transaction will always either succeed or fail. If it succeeds it may be legally binding for the two partners. 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 transaction upon failure.

3. Business Document Flows

A business transaction is realized as Business Document Flows between the requesting and responding activities. 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 a specification schema instance 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 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 specification schema provides a set of unambiguous semantics within which to specify transactions and collaborations. Within these semantics the using community have flexibility to specify very different transactions and collaborations. The use of patterns combines this flexibility with a consistency that facilitates faster design, faster implementation, and enables generic processing.

A set of parameters determine the semantics of a business transaction.

A set of predefined transaction patterns can be found in UMM.

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

6.2 How to use the ebXML Business Process Specification Schema

The ebXML Specification Schema should be used wherever software is being specified to Business Collaborations.

The ebXML Specification Schema is used to specify the business process related configuration parameters for configuring software to execute these collaborations

This section discusses

- How the Specification Schema fits in with other ebXML specifications.
- How to use the 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 CPP/CPA.
- Run-time transaction and collaboration semantics that the Specification Schema specifies and the Business Service Interface (BSI) is expected to manage.

6.3 How Specification Schema is used with other ebXML specifications

The ebXML 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.

The business documents are defined at the intersection between the business process model and the information model. At ebXML that means that the Business Documents are assembled from lower level information structures known as 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 model and the information model become the basis against which partners can make agreements on conducting electronic business with each other.

Specifically The ebXML Specification Schema is intended to provide the business process specification for the formation of a partner Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA).

The actual configuration of the Business Service Interface (BSI) is done according to the ebXML Collaboration Protocol Profile specification.

This process can be illustrated as follows:

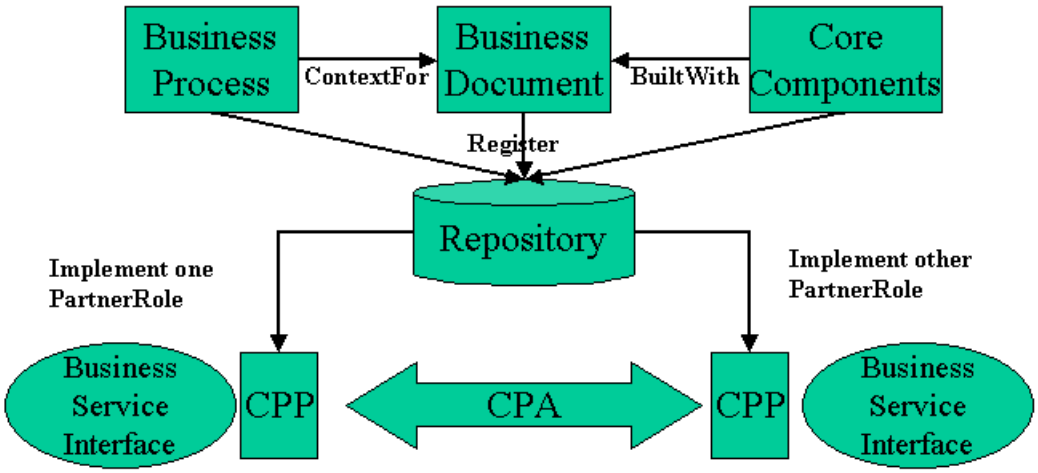


Figure 4. Specification Schema and other ebXML Specifications

The Specification Schema allows the specification of business processes. The resulting business process specification is stored in the repository, and provides context for document definition. It then provides the business parameters for CPP and CPA creation.

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.

A user would use a UML tool to create a model instance against the Specification Schema, and would then use the production rules to produce the XML version of the model, compliant with the DTD version of the specification schema.

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

In either case, the XML version gets stored in the ebXML repository and registered in the ebXML registry for future retrieval when implementers want to establish trading partner Collaboration Protocol Profile and Agreement. At that

point the XML document, or the relevant parts of it, are simply imbedded in or referenced by the CPP and CPA XML documents.

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 Specification Schema modeling relationships by building a complete Multiparty Collaboration from the bottom up, as follows:

1. Specify a Business Transaction
2. Specify Document Flows 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 model from the bottom up, the Specification Schema very much is intended for modeling from the top down, re-using existing lower level content as much as possible.

6.4.1 Specify a Business Transaction and its Document Flow

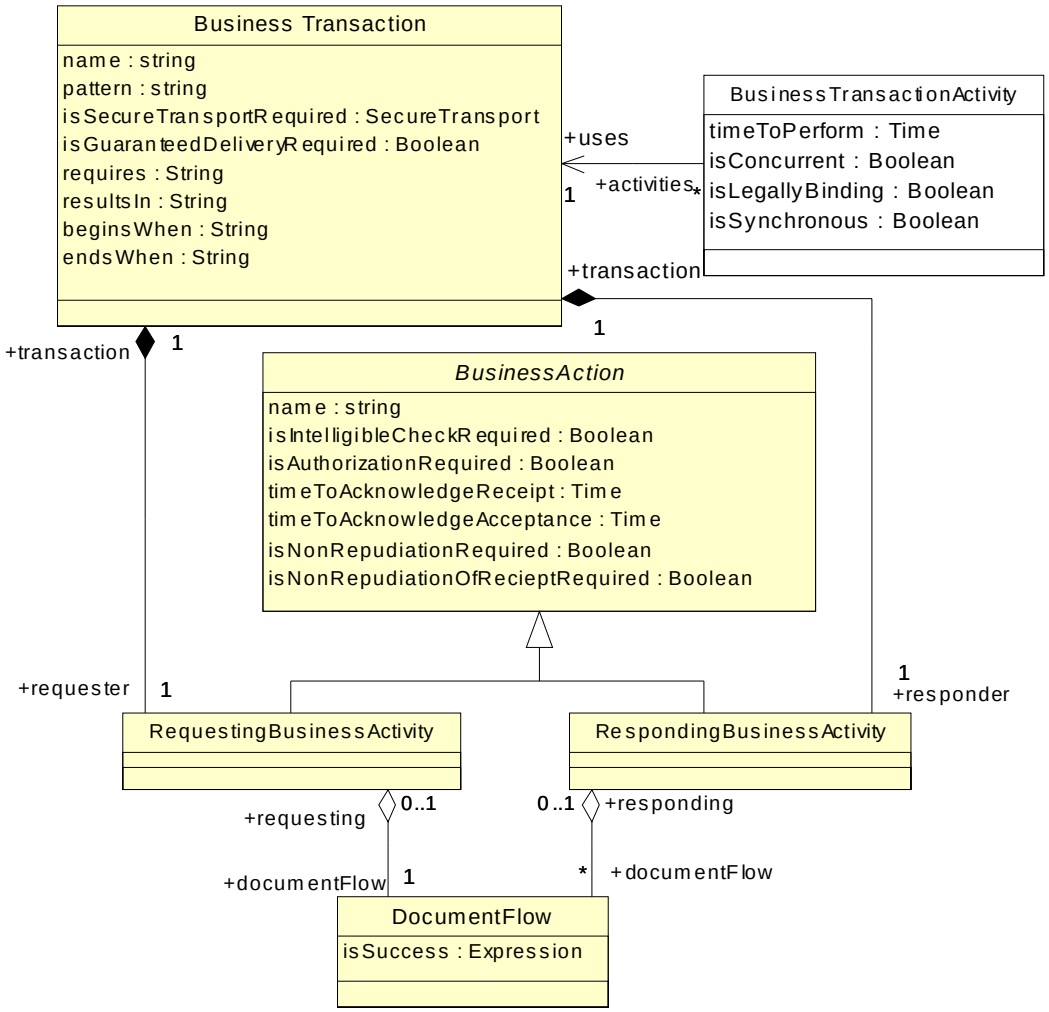


Figure 5. Semantics of a Business Transaction

6.4.1.1 Key Semantics of a Business Transaction

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

A business transaction consists of a Requesting Business Activity, a Responding Business Activity, and one or two Document Flows between them.

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="">
            <DocumentFlow isSuccess="true"
              documentType 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 can be shown as follows:

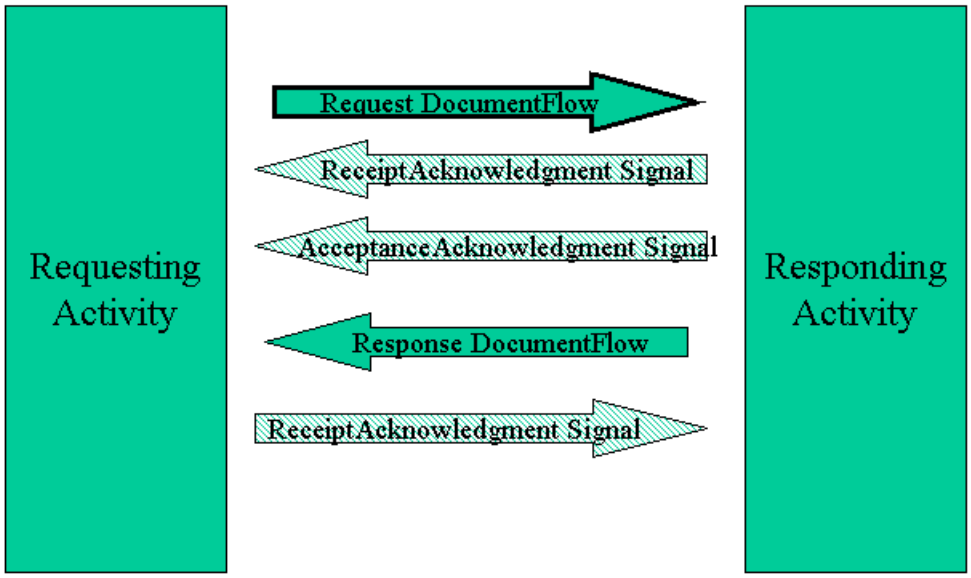


Figure 6. Possible document flows and signals and their sequence

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 sent is part of the selection of an interaction pattern for the Business Transaction. These business signals have specific business purpose and are separate from lower protocol and transport signals.

Receipt acknowledgement business signal. The UN/EDIFACT model Trading Partner Agreement (TPA) suggests that a partners should agree on the point at which a message can be "said" to be properly received and this point should be when a receiving partner can "read" a message[1]. The property ***isIntelligibleCheckRequired*** allows partners to agree that a message should be "readable" before its receipt is verified[2].

Acceptance Acknowledgement business signal. The UN/EDIFACT model TPA suggests that partners should agree on the point at which a

[1] The UN/ECE defines this to be the point after which a message passes a structure/ schema validity check. This is not a necessary condition for verifying proper receipt, only accessibility is.
[2]

message can be "said" to be accepted for business processing and this point should be after the contents of a business document have passed a business rule validity check.[3]

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/ISO 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">
            <DocumentFlow isSuccess="true"
                  documentType="Purchase Order"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name=""
                  isNonRepudiationRequired="true"
            timeToAcknowledgeReceipt=”P5D">
            <DocumentFlow isSuccess="true"
                  documentType="PO Acknowledgement"/>
            </DocumentFlow>
      </RespondingBusinessActivity>
</BusinessTransaction>
```

6.4.1.4 Specifying Document Flows

Request Document Flows and Response Document Flows contain Business Documents that pertain to the Business Transaction. The model for this is shown below. Business Documents all have unique structures. Business signals, however always have the same structure, defined once and for all as part of the common modeling elements.

[3] This is the convention specified for RosettaNet and UN/CEFACT N90 commercial transactions.

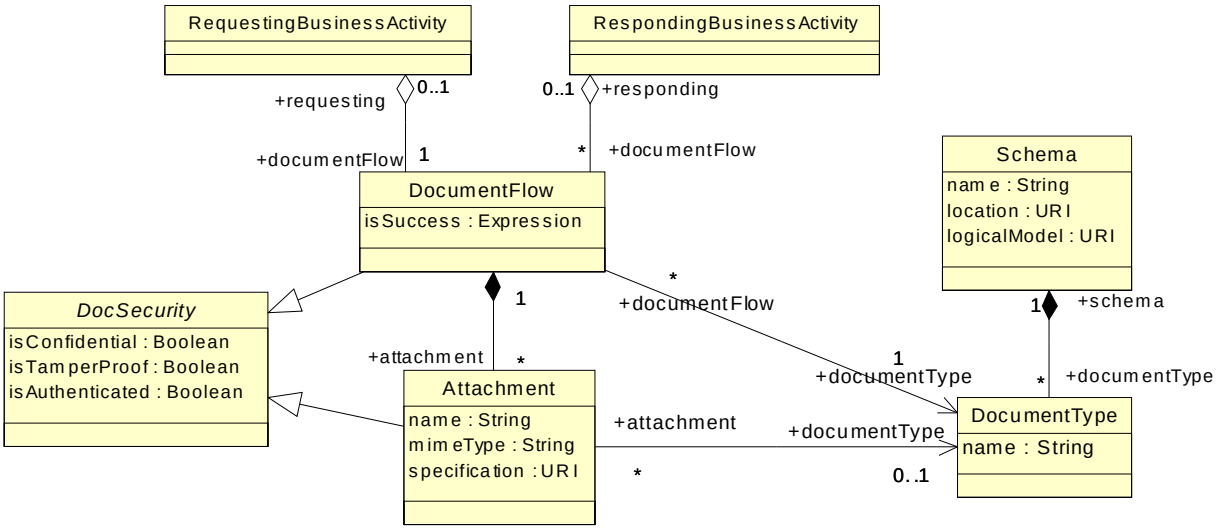


Figure 7. UML semantics of a Document Flow

Document Flows are named. There is always only one named Document Flow for a Requesting Activity. There may be zero, one, or many named Document Flows for a Responding Activity. For example, the Response Document Flows 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 Flow represents the flow of documents between the activities. Each Document Flow carries exactly one primary Business Document. The document is specified in terms of its DocumentType.

A Document Type is defined in a Schema. This may be an ebXML Schema, or a Schema supplied by an outside source.

A Document Flow 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 TRP message.

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 schema, and the document types are fully qualified with the schema name.

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

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

6.4.2 Specify a Binary Collaboration

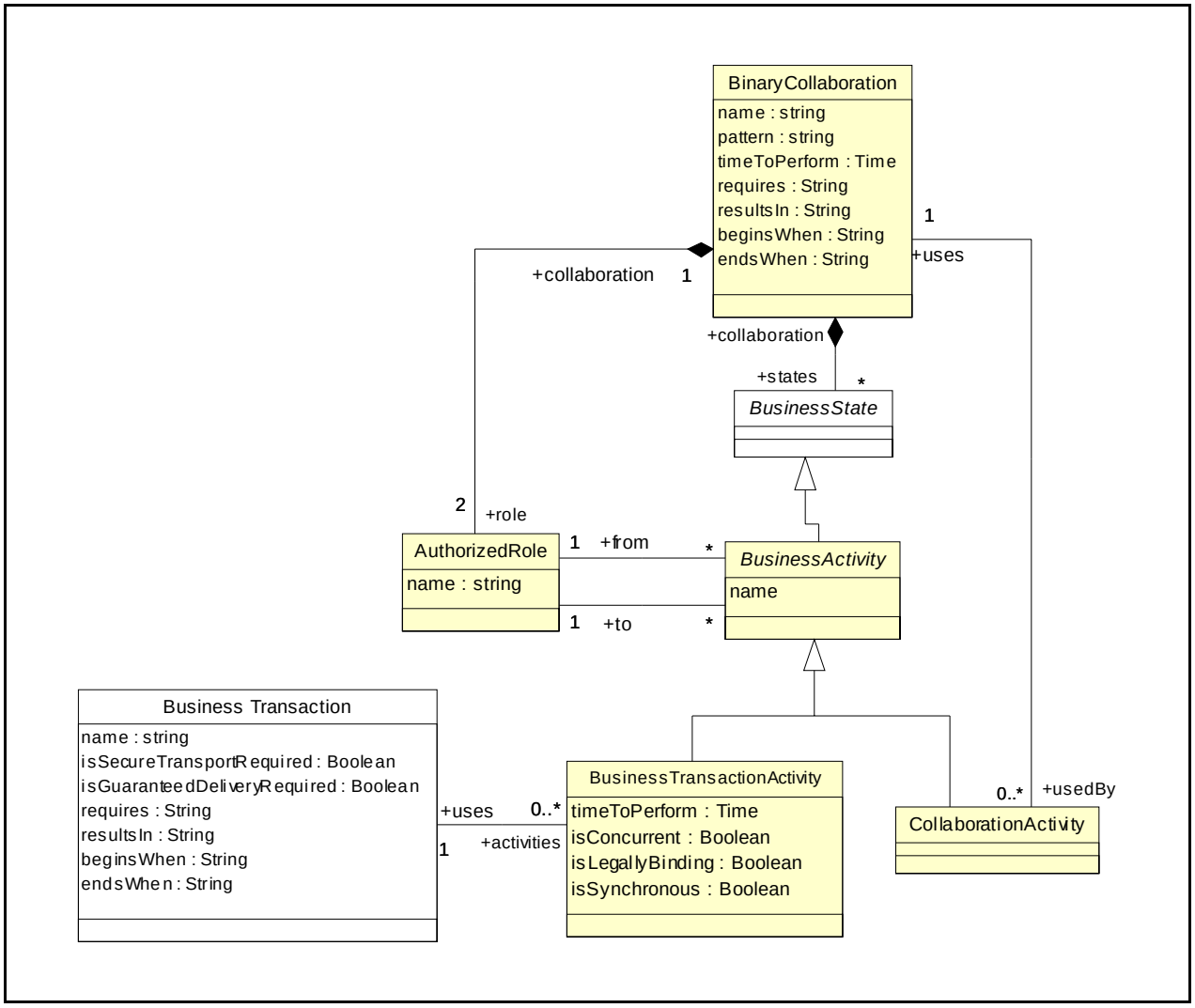


Figure 8. UML semantics of a Binary Collaboration

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 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 must be referenced through Business Transaction Activities contained within a Binary Collaboration.

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. Under the developing laws of electronic signatures, a declaration of intent to be bound is a key element in establishing the legal equivalence of an electronic message to a signed physical writing. However, in ebXML, the presence or absence of an electronic signature cannot indicate legally binding assent, because it is reserved for use as an assurance of sender identity and message integrity.

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.

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

As in EDI, the ebXML standard assumes that Business Transactions are intended by the trading parties to be binding unless otherwise indicated. 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. That value indicates to the trading parties that the activities conducted in that transaction are only intended as a proposal or test, and that they may not rely on or seek to enforce the other's activity."

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

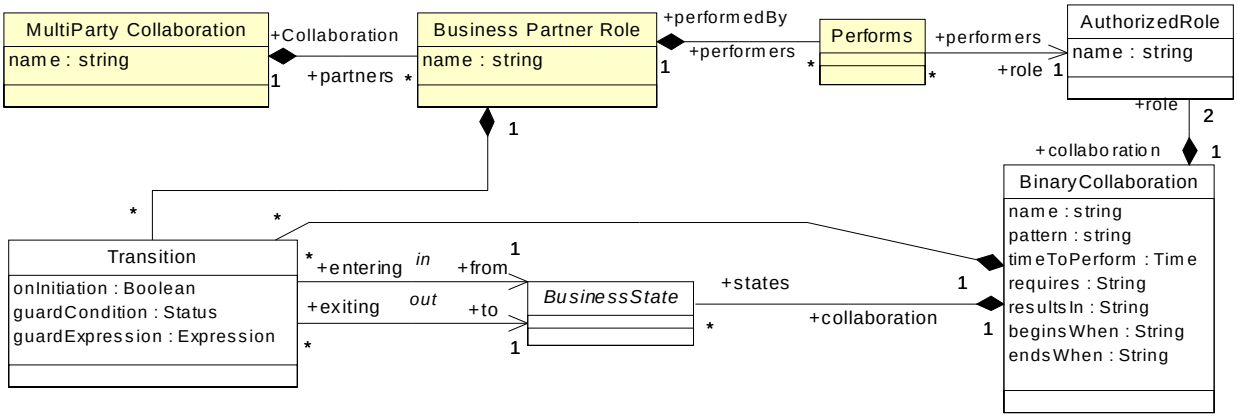


Figure 9. UML semantics of a MultiParty Collaboration

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 Transactions 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
                  binaryCollaboration="Firm Order”
            authorizedRole="buyer"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="Retailer">
            <Performs
                  binaryCollaboration="Firm Order”
            authorizedRole="seller"/>
            <Performs
              binaryCollaboration="Product Fulfillment”
            authorizedRole="buyer"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="DropShip Vendor">
            <Performs
              binaryCollaboration="Product Fulfillment”
            authorizedRole="seller"/>
      </BusinessPartnerRole>
</MultiPartyCollaboration>
```

6.4.4 Specify a Choreography

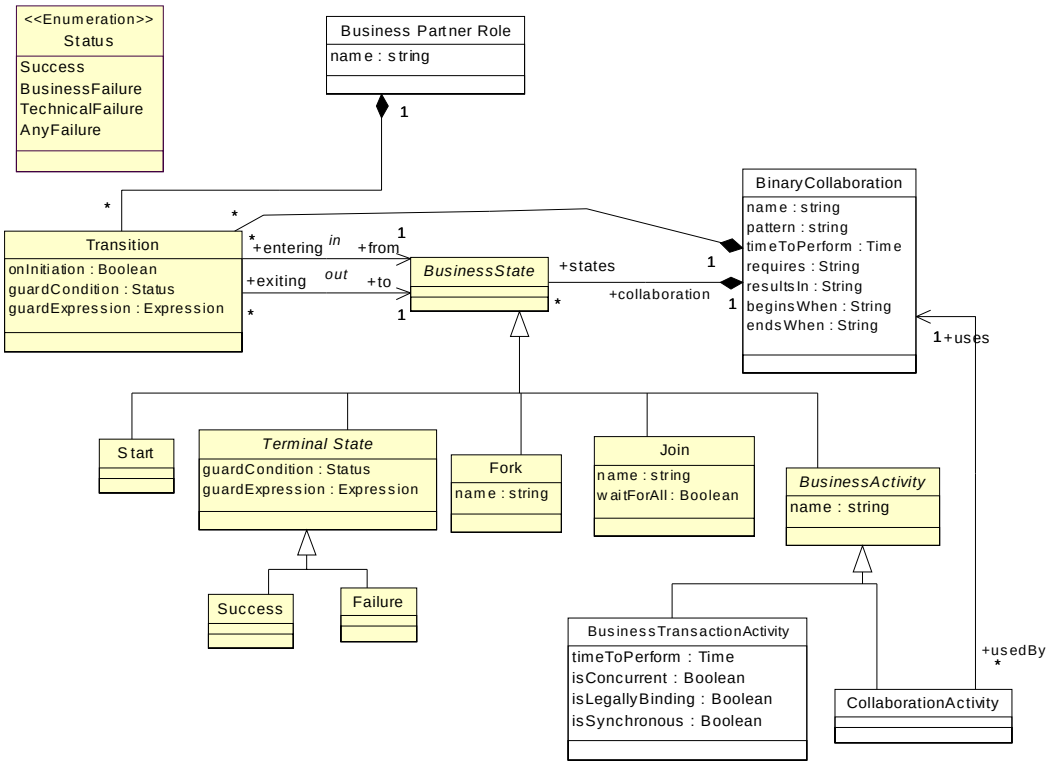


Figure 10. UML semantics of a Choreography

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 Flow 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 the BusinessTransactionActivity 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.

Concurrency 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 the parameter that governs this. It is at 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.

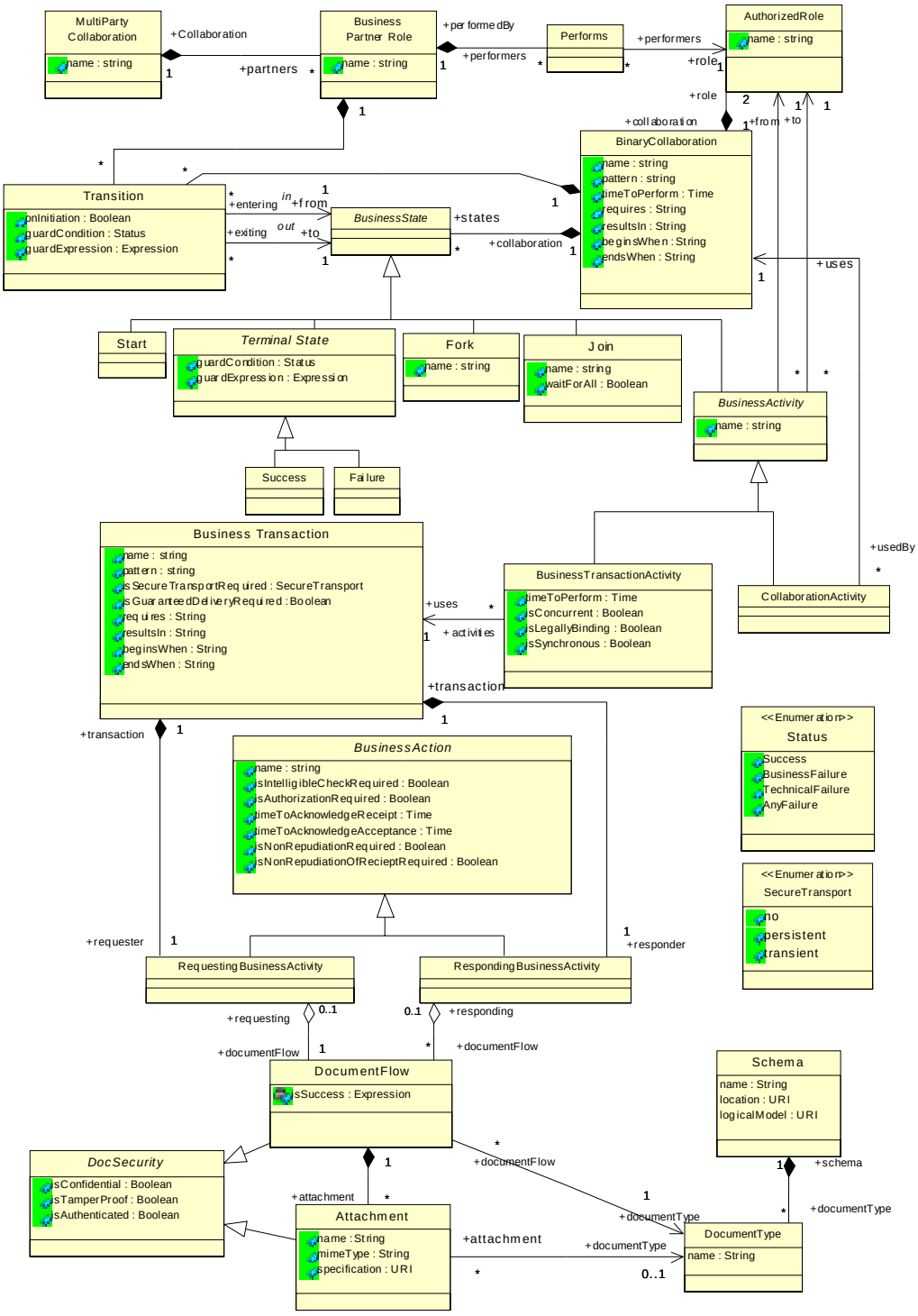
```
            <MultiPartyCollaboration name="DropShip">
                  <BusinessPartnerRole name="Customer">
                        <Performs
                              binaryCollaboration="Firm Order”
                        authorizedRole="buyer"/>
                  </BusinessPartnerRole>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

6.4.5 The whole model

The following picture shows the above semantics collectively as a UML class diagram. This diagram contains the whole UML version of the Specification Schema

982

Figure 11. Overall Specification Schema as UML class diagram



983

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.

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. The response (if any) transitions control back to the requesting role. If no response is sent then control transitions back to the requesting role based on the receipt of a business signal. Regardless which combination of receiptAcknowledgement and/or acceptanceAcknowledgement is chosen and/or Response Document Flow is chosen, they always flow in the sequence just listed, and the last one always transfers control back to the requesting activity.
9. All business transactions succeed or fail. Success or failure depends on:
    a. The receipt or non-receipt of the response or business signals
    b. The occurrence of time-outs
    c. The occurrence of a business exception
    d. The occurrence of a control exception

10. Upon receipt of the Response Document Flow 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.

6.5.1 Defining and using transaction and interaction patterns

Even though all business transactions are governed by very precise semantics, business transactions can be carried out in many distinctly and succinctly Specified patterns and with different security and exchange characteristics.

The groups of specification elements that determine the exact pattern of a business transaction are:

- Transaction Patterns and Time-out parameters
- Security parameters.
- Reliability parameters
- Synchronous or asynchronous

6.5.2 Transaction Patterns

Transaction patterns are based on whether a response is required and whether receiptAcknowledgement and/or acceptanceAcknowledgement are required.

The way to specify that a receiptAcknowledgement is required is to set the parameter timeToAcknowledgeReceipt to any value other than blank.

The way to specify that a acceptanceAcknowledgement is required is to set the parameter timeToAcknowledgeAcceptance to any value other than blank.

So these two 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.

These could then be given pattern names and business process definition tools could facilitate the application of named patterns to actual business transaction definitions.

UMM provides a set of already specified transaction patterns for re-use.

6.5.3 Parameters required for CPP/CPA

The specification schema provides parameters that can be used to specify certain levels of security and reliability. The 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 Delivery Channel selection, that is the selection of the transport facility and transport characteristics, depending on security and reliability parameters.

The other kind of parameter determines the service level or capability of the BSI itself.

6.5.4 Delivery Channel selection parameters:

6.5.4.1 Document security:

Each business document being transported, even if many are collected in the same message, can be specified as:

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.

6.5.4.2 Document Flow security:

Each message can be specified to require secure transport.

Parameter	Delivery Channel requirement
IsSecureTransportRequired	This means that Business Documents transferred must be treated as per parameters *isConfidential,*

	isTamperProof, isAuthenticated, above

IsSecureTransportRequired can have three values:

"Transient" means *isConfidential, isTamperProof, isAuthenticated,* on the wire.

"Persistent" means *isConfidential, isTamperProof, isAuthenticated,* until delivered into the application space.

"No" means no security required

The value of isSecureTransportRequired and the document level values of *isConfidential, isTamperProof, isAuthenticated* combine to produce the value that calls for the strongest security. E.g. if isSecureTransportRequired = 'Transient', then all documents will be treated as *isConfidential, isTamperProof, isAuthenticated* on the wire even if their individual settings are 'no'. Likewise a document that requires security will be treated with that security, even if isSecureTransportRequired is 'no'.

6.5.5 Reliability

A parameter at the transaction level states whether guaranteed delivery required.

Parameter	Delivery Channel requirement
IsGuaranteedDeliveryRequired	This means that Business Documents transferred guaranteed to be delivered

This is a parameter in the business transaction

6.5.6 Synchronous or Asynchronous

Need proper definition possibly in terms of blocking

Parameter	Delivery Channel requirement
isSynchronous	Synchronous response

A business transaction may be implemented as either a synchronous or an asynchronous flow of control between the two activities. The specification of synchronous vs. asynchronous is

part of the interaction pattern specification for a business transaction.

A partner role that initiates an asynchronous business transaction does not need to receive any business signals. A partner role that initiates a synchronous business transaction must be able to receive business signals and must block until the flow of control is returned. This should not preclude the initiation and execution of multiple concurrent business transactions, however.

6.5.7 BSI service level parameters:

6.5.7.1 Action 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.

6.5.7.2 Non-Repudiation

Trading partners may wish to conduct legally enforceable business transactions over ebXML. Parties 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 messages.

There are two kinds of non-repudiation protocol available under this document.

One imposes a duty on each party to save copies of all [Business Documents], each on their own side, i.e., requestor saves his request, responder saves his response. This is the isNonRepudiationRequired in the requesting or responding activity. It is logically equivalent to a request that the other trading partner maintain an audit trail.

The other requires the responder to send a signed copy of the receipt, which the requestor then saves. This is the isNonRepudiationOfReceiptRequired in the requesting business activity.

NonRepudiationOfReceipt is tied to the ReceiptAcknowledgement, in that it requires it to be signed. So one cannot have NonRepudiationOfReceipt without ReceiptAcknowledgement required, and the timeToAcknowledgeReceipt applies to both, i.e. if NonRepudiationOfReceipt is true and a signed receipt is not returned within timeToAcknowledgeReceipt, the transaction is null and void.

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

6.6 Run time 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
4. Detection of failed completion of a transaction
    a. Detection of time-outs
    b. Detection of exceptions

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 is 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 each of the response types. If the time-out occurs before the required response 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 requesting business document is sent. The timer values must comply with the well-formedness rules for timer values.

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.

6.6.3 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.

6.6.4 Business Protocol Exceptions (a.k.a. ProcessException)

A *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 the semantics of message elements and the validity of the request itself and 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 responding role that throws a business protocol exception signals the exception back to the requesting role and then terminates the business transaction. A requesting role that throws a business protocol exception terminates the transaction and then sends as a separate transaction a notification revoking the offending business document request. The requesting role does not send a business exception signal to the responding role.

If any business exceptions (includes negative receipt and acceptance acknowledgements) are signaled then the business transaction must terminate.

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 Specification Schema May Be Implemented

The ebXML Specification Schema should be used wherever software is being specified to perform a role in an ebXML business collaboration. Specifically The ebXML 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 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 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:

NONE

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 Roles in each of the Binary Collaborations that make up the Multiparty Collaboration.

Tagged Values:

name. The name of the roles 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

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

role Performs is the use of an AuthorizedRole within a multiparty collaboration

Wellformedness Rules:

For every Performs performing an AuthorizedRole there must be a Performs that performs the opposing AuthorizedRole, otherwise the MultiParty Collaboration is not complete.

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. The name of the role within the business transaction

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 transaction rollback is required.

Tagged Values:

timeToPerform.	The time allowed to complete the binary collaboration
requires	A description of a state external to this collaboration/transaction that is required before this collaboration/transaction to conclude.
resultsIn	A description of a state that does not exist before the execution of this collaboration/transaction but will exists as a result of the execution of this collaboration/transaction.
beginsWhen	A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to commence.
endsWhen	A description of an event external to this collaboration/transaction that normally causes this collaboration/transaction to conclude.
Pattern	The optional name of the 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.

Tagged Values:

name.	The name of the activity 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. The business transaction activity can be executed more than once if the ***isConcurrent*** property is ***true***.

Tagged Values:

timeToPerform.	Both partners agree to perform a business transaction activity within a specific duration. The originating partner must send a failure notification to a responding partner on timeout. A responding partner simple terminates its activity. The time to perform is the duration from the time a business transaction activity initiates the first business transaction until there is a transition back to the initiating business transaction activity. Both partners agree that the business signal document or business action document specified as the document to return within the time to perform is the “Acceptance Document” in an on-line offer/acceptance contract formation process.
isConcurrent.	If the business transaction activity is concurrent then more than one business transaction can be open at one time. If the business transaction

activity is not concurrent then only one business transaction activity can be open at one time.

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

isSynchronous The Business Transaction is intended to be performed by this activity in a synchronous manner

Associations:

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

Wellformedness Rules:

NONE

7.1.8 CollaborationActivity

A collaboration activity defines the use of one binary collaboration within another.

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

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 an in the distribution of product announcements.

Tagged Values:

name The name of the Business Transaction.

isSecureTransportRequired.

Both partners must agree to exchange business information using a secure Delivery Channel. The following security controls ensure that business document content is protected against unauthorized disclosure or modification and that business services are protected against unauthorized access. This is a point-to-point security requirement. Note that this requirement does not protect business information once it is off the network and inside an enterprise. The following are requirements for secure transport Delivery Channels.

Authenticate sending role identity – Verify the identity of the sending role (employee or organization) that is initiating the role interaction (authenticate).

Authenticate receiving role identity – Verify the identity of the receiving role (employee or organization) that is receiving the role interaction.

Verify content integrity – Verify the integrity of the content exchanged during the role interaction i.e. check that the content has not been altered by a 3rd party.

Maintain content confidentiality – Confidentiality ensures that only the intended, receiving role can read the content of the role interaction

isGuaranteedDeliveryRequired. Both partners must agree to use a transport that guarantees delivery

requires A description of a state external to this collaboration/transaction that is required before this collaboration/transaction to conclude.

resultsIn A description of a state that does not exist before the execution of this collaboration/transaction but will exists as a result of the execution of this collaboration/transaction.

beginsWhen A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to commence.

endsWhen A description of an event external to this collaboration/transaction that normally causes this collaboration/transaction to conclude.

pattern The optional name of the pattern that this business transaction is based on

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:

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-reputable. A receiving partner must send notification of failed business control (possibly revoking a contractual offer) if a responding partner has not properly delivered their business document.

Non-repudiation of receipt provides 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.

timeToAcknowledgeAcceptance The time a receiving role has to non-substantively acknowledge business acceptance of a business document.

timeToAcknowledgeReceipt The time a receiving role has to acknowledge receipt of a business document.

Associations:

NONE

Wellformedness Rules:

NONE

7.2.3 RequestingBusinessActivity

A RequestingBusinessActivity is a business activity that is performed by a role requesting commerce from another role. It specifies the Document Flow which will carry the request.

Tagged Values:

NONE, except as inherited from Business Action

Associations:

transaction A requesting activity is performed in exactly one business transaction

documentFlow A responding activity sends at most one Document Flow

Wellformedness Rules:

NONE

7.2.4 RespondingBusinessActivity

A RespondingBusinessActivity is a business activity that is performed by a role responding to another business role's request for commerce.

It specifies the Document Flow which will carry the request.

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

Tagged Values:

NONE, except as inherited from Business Action

Associations:

transaction	A responding activity is performed in exactly one business transaction
documentFlow	A responding activity sends at most one Document Flow

Wellformedness Rules:

NONE

7.3 Document Flow

7.3.1 Document Flow

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

Tagged Values:

isSuccess	An expression whose evaluation results in TRUE or FALSE as the determination of whether this DocumentFlow should be considered the successful initiation or conclusion of a BusinessTransaction.

Associations:

requesting	A Document Flow is sent by at most one requesting activity
responding	A Document Flow is sent by at most one responding activity
documentType	A Document Flow contains only one Document
attachment	A Document Flow contains an optional set of attachments to the document

Wellformedness Rules:

A Document Flow cannot be sent by both a requesting and a responding activity

.

7.3.2 DocumentType

DocumentType is a generic name of a document. The definition of the document can be found in the associated Schema. Associated with the document can optionally be a set of attachments

Tagged Values:

name	The name of the document type

Associations:

schema	A document type is in at most one schema
documentFlow	A document type can be in multiple document flows
attachment	A document type can specify many attachments

Wellformedness Rules:

NONE

7.3.3 Schema

A Schema is a collection of Document Definitions. The Schema 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 Schema. Typically this would be an ebXML core component context model.

Tagged Values:

Name	The name of the schema
location	Reference to an external source of the schema definition
logicalModel	Reference is to where the logical model is

Associations:

documentType	A schema defines many document types

Wellformedness Rules:

NONE

7.3.4 Attachment

Attachment is an optional attachment to a DocumentType in a Document Flow

Tagged Values:

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

Associations:

documentFlow	An Attachment is in exactly one document flow
documentType	An Attachment is of at most one document type. If it is not of a defined document type, the mime type and spec will be the only indication of its type

Wellformedness Rules:

An attachment is always related to the primary document an a document flow

7.4 Choreography within Collaborations.

7.4.1 BusinessState

A business state is any state that a binary collaboration can be in. Some business states are a snapshot right after or right before an activity, others are action states that denote the state of being in an activity.

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	Is used to specify this is a nested BusinessTransactionActivity and a second transaction is performed before returning to the this 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 document in the most recent DocumentFlow.

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.

Tagged Values:

NONE

Associations:

NONE

Wellformedness Rules:

NONE

7.4.4 TerminalState

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

Tagged Values:

NONE

Associations:

NONE

Wellformedness Rules:

NONE

7.4.5 Success

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

Tagged Values:

NONE.

Associations:

NONE

Wellformedness Rules:

Every activity Binary Collaboration should have at least one success

7.4.6 Failure

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

Tagged Values:

None.

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.

Tagged Values:

None

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.

Tagged Values:

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

7.4.9 Guard

The condition under which a transition may happen.

Tagged Values:

Precondition	A condition referring to the status of the previous document flow: Success, or the outcome of the previous transaction activity: BusinessFailure, TechnicalFailure

Associations:

Transition	*The transition(s) that this guard governs*

Wellformedness Rules:

Guards must refer only to the immediately prior document flows and/or the immediately prior transaction activity

7.5 Definition and Scope

The ebXML Specification Schema should be used wherever software is being specified to perform a role in an ebXML binary collaboration. Specifically The ebXML 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 Specification Rules

The following rules are used to constrain the values of the parameters of the elements in the Specification Schema.

7.6.1 Well-formedness Rules

The following well-formedness rules apply :

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 that in turn is the input to a responding business activity.

[14] There must be zero or one output Document Flow from a requesting 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 Specification Schema – (DTD)

In this section we describe the DTD version of the Specification Schema. This discussion includes

- A listing of the DTD itself
- A table listing all the elements found in the DTD with definitions and parent/child relationships
- A table listing all the attributes found in the DTD with definitions and parent element relationships
- A table listing all the elements found in the DTD, each with a cross reference to the corresponding class in the UML version of the specification schema
- An example using the DTD
- Rules about namespaces

Additionally, following this section, a section will describe common modeling elements, including data types, and DTDs for common business signals. And a section after that describes the production rules we intend to follow for creating XML documents from a model instance against the UML specification schema.

8.1 DTD

This is the Specification Schema DTD:

```
<? version="1.0" encoding="UTF-8"?>
<!--========================================================== -->
<!--  Editor:   Kurt Kanaskie (Lucent Technologies)           -->
<!--  Version:  Version .99                                   -->
<!--  Updated:  March 20, 2001                                -->
<!--                                                          -->
<!--  Public Identifier:                                      -->
<!-- "-//ebXML//DTD Process Specification ver .99//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 N90 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 Minute   -->
<!-- ========================================================== -->

<!ELEMENT EbXmlProcessSpecification (Documentation*, Include*,
Package*)>
<!ATTLIST EbXmlProcessSpecification
      name CDATA #REQUIRED
      version CDATA #REQUIRED
      uuid CDATA #REQUIRED
>
<!ELEMENT Documentation (#PCDATA)>
<!ATTLIST Documentation
      uri CDATA #IMPLIED
>

<!ELEMENT Include EMPTY>
<!ATTLIST Include
      name CDATA #REQUIRED
      version CDATA #REQUIRED
      uuid CDATA #REQUIRED
      uri CDATA #REQUIRED
>
<!ELEMENT Package (Documentation*,
             (Package | BinaryCollaboration |
MultiPartyCollaboration | BusinessTransaction | Schema)*)>
<!ATTLIST Package
      name CDATA #REQUIRED
>
<!-- Model requires 2 Authorized roles -->
<!ELEMENT BinaryCollaboration (Documentation*, AuthorizedRole,
AuthorizedRole,
                      (Documentation* | Start | Transition |
Success | Failure | Fork | Join | BusinessTransactionActivity |
CollaborationActivity)*)>
<!ATTLIST BinaryCollaboration
      name CDATA #REQUIRED
      pattern CDATA #IMPLIED
      beginsWhen CDATA #IMPLIED
      endsWhen CDATA #IMPLIED
      requires CDATA #IMPLIED
      resultsIn CDATA #IMPLIED
      timeToPerform CDATA #IMPLIED
>
<!-- fromBusinessState and toBusinessState are the names of a
     BusinessTransactionActivity or CollaborationActivity
     both are logical targets for from/to -->
<!-- "guardExpression" is an expression that results in a boolean
true or false -->
<!ELEMENT Transition (Documentation*)>
<!ATTLIST Transition
```

```
      onInitiation (true | false) "false"
      fromBusinessState CDATA #IMPLIED
      toBusinessState CDATA #IMPLIED
      guardCondition (Success | BusinessFailure |
TechnicalFailure | AnyFailure ) #IMPLIED
      guardExpression CDATA #IMPLIED
>
<!ELEMENT Start EMPTY>
<!ATTLIST Start
      toBusinessState CDATA #REQUIRED
>
<!ELEMENT Success EMPTY>
<!ATTLIST Success
      fromBusinessState CDATA #REQUIRED
      guardCondition (Success | BusinessFailure |
TechnicalFailure | AnyFailure ) #IMPLIED
      guardExpression CDATA #IMPLIED
>
<!ELEMENT Failure EMPTY>
<!ATTLIST Failure
      fromBusinessState CDATA #REQUIRED
      guardCondition (Success | BusinessFailure |
TechnicalFailure | AnyFailure ) #IMPLIED
      guardExpression CDATA #IMPLIED
>

<!ELEMENT Fork EMPTY>
<!ATTLIST Fork
      name CDATA #REQUIRED
>
<!ELEMENT Join EMPTY>
<!ATTLIST Join
      name CDATA #REQUIRED
      waitForAll (true | false) "true"
>
<!-- fromAuthorizedRole and toAuthorizedRole are fully qualified
names -->
<!ELEMENT BusinessTransactionActivity (Documentation*)>
<!ATTLIST BusinessTransactionActivity
      name CDATA #REQUIRED
      businessTransaction CDATA #REQUIRED
      fromAuthorizedRole CDATA #REQUIRED
      toAuthorizedRole CDATA #REQUIRED
      isConcurrent (true | false) "false"
      isLegallyBinding (true | false) "true"
      isSynchronous (true | false) "false"
      timeToPerform CDATA #IMPLIED
>
<!ELEMENT CollaborationActivity (Documentation*)>
<!ATTLIST CollaborationActivity
```

```
      name CDATA #REQUIRED
      fromAuthorizedRole CDATA #REQUIRED
      toAuthorizedRole CDATA #REQUIRED
      binaryCollaboration CDATA #REQUIRED
>
<!ELEMENT BusinessTransaction (Documentation*,
RequestingBusinessActivity, RespondingBusinessActivity)>
<!ATTLIST BusinessTransaction
      name CDATA #REQUIRED
      pattern CDATA #IMPLIED
      beginsWhen CDATA #IMPLIED
      endsWhen CDATA #IMPLIED
      isSecureTransportRequired (no | persistent | transient)
"no"
      isGuaranteedDeliveryRequired (true | false) "false"
      requires CDATA #IMPLIED
      resultsIn CDATA #IMPLIED
>
<!ELEMENT RequestingBusinessActivity (Documentation*,
DocumentFlow)>
<!ATTLIST RequestingBusinessActivity
      name CDATA #REQUIRED
      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*,
DocumentFlow*)>
<!ATTLIST RespondingBusinessActivity
      name CDATA #REQUIRED
      isAuthorizationRequired (true | false) "false"
      isIntelligibleCheckRequired (true | false) "false"
      isNonRepudiationReceiptRequired (true | false) "false"
      isNonRepudiationRequired (true | false) "false"
      timeToAcknowledgeAcceptance CDATA #IMPLIED
      timeToAcknowledgeReceipt CDATA #IMPLIED
>
<!ELEMENT Schema (Documentation*, DocumentType*)>
<!ATTLIST Schema
      name CDATA #REQUIRED
      location CDATA #IMPLIED
      logicalModel CDATA #IMPLIED
>
<!ELEMENT DocumentType (Documentation*)>
<!ATTLIST DocumentType
      name CDATA #REQUIRED
>
```

```
<!ELEMENT DocumentFlow (Documentation*, Attachment*)>
<!-- "isSuccess" is an expression that results in a boolean true
or false -->
<!-- documentType is the fully qualified name -->
<!ATTLIST DocumentFlow
      documentType CDATA #REQUIRED
      isSuccess CDATA #REQUIRED
      isAuthenticated (true | false) "false"
      isConfidential (true | false) "false"
      isTamperProof (true | false) "false"
>
<!ELEMENT Attachment (Documentation*)>
<!ATTLIST Attachment
      name CDATA #REQUIRED
      mimeType CDATA #REQUIRED
      documentType CDATA #IMPLIED
      specification CDATA #IMPLIED
      isConfidential (true | false) "false"
      isTamperProof (true | false) "false"
      isAuthenticated (true | false) "false"
>
<!ELEMENT AuthorizedRole (Documentation*)>
<!ATTLIST AuthorizedRole
      name CDATA #REQUIRED
>
<!ELEMENT MultiPartyCollaboration (Documentation*,
BusinessPartnerRole*)>
<!ATTLIST MultiPartyCollaboration
      name CDATA #REQUIRED
>
<!ELEMENT BusinessPartnerRole (Documentation*, Performs*,
Transition*)>
<!ATTLIST BusinessPartnerRole
      name CDATA #REQUIRED
>
<!-- authorizedRole is the fully qualified name
BinaryCollaboration@name/AuthorizedRole@name -->
<!ELEMENT Performs (Documentation*)>
<!ATTLIST Performs
      authorizedRole CDATA #REQUIRED
>
```

8.2 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.

a. Attachment

XML Element Name: Attachment

DTD Declaration:

```
<!ELEMENT Attachment (Documentation*)>
<!ATTLIST Attachment
      name CDATA #REQUIRED
      mimeType CDATA #REQUIRED
      Spec CDATA #REQUIRED
      isConfidential (true | false) "false"
      isTamperProof (true | false) "false"
      isAuthenticated (true | false) "false">
```

Definition:

An optional attachment to a DocumentType in a DocumentFlow.

Parent Elements:

- DocumentFlow

Attributes:

Attribute Name	Definition	Default Value
isAuthenticated	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.	**false** {true, false}
isConfidential	The information entity is encrypted so that unauthorized parties cannot view the information	**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.	**false** Valid values {true, false}
mimeType	Defines the valid MIME (Multipurpose Internet Mail Extensions) type of this Attachment	**Required input.** Example: 'application/pdf'
name	Defines the name of the attachment.	**Required input.**
Spec	A reference to an external source of description of this attachment.	**Required input.**

b. Authorized Role

XML Element Name: AuthorizedRole

DTD Declaration:

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

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.**

c. Binary Collaboration

XML Element Name: BinaryCollaboration

DTD Declaration:

```
<!ELEMENT BinaryCollaboration (Documentation*,
AuthorizedRole, AuthorizedRole,
(Documentation* | Start | Transition | Success |
Failure | Fork | Join | BusinessTransactionActivity |
CollaborationActivity)*)>
<!ATTLIST BinaryCollaboration
      name ID #REQUIRED
      beginsWhen CDATA #IMPLIED
      endsWhen CDATA #IMPLIED
      requires CDATA #IMPLIED
      resultsIn 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 transaction rollback is required.

Parents:

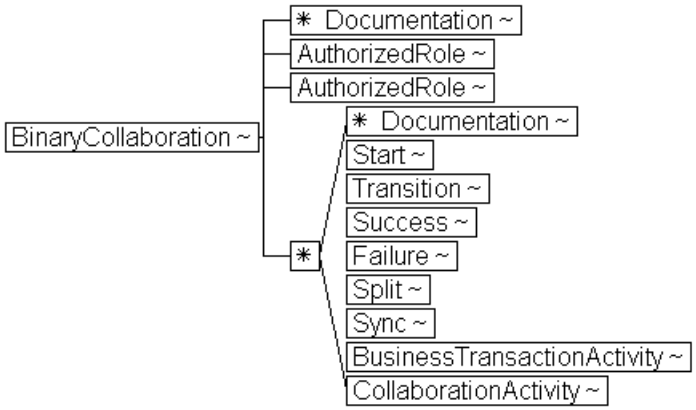
- Package

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.
beginsWhen	A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to commence.	Optional Input.

endsWhen	A description of an event external to this collaboration/transaction that normally causes this collaboration/transaction to conclude.	Optional Input.
requires	A description of a state external to this collaboration/transaction that is required before this collaboration/transaction to conclude.	Optional Input.
resultsIn	A description of a state that does not exist before the execution of this collaboration/transaction but will exists as a result of the execution of this collaboration/transaction.	Optional Input..
timeToPerform	The period of time, starting upon initiation of the first activity within which this entire collaboration must conclude.	Optional Input.

Hierarchical Model:



c: Business Partner Role

Element Name: BusinessPartnerRole

DTD Declaration:

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

Definition:

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

Parents:

- MultiPartyCollaboration

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.

d. Business Transaction

Element Name: BusinessTransaction

Content Model:

```
<!ELEMENT BusinessTransaction (Documentation*,
RequestingBusinessActivity, RespondingBusinessActivity)>
<!ATTLIST BusinessTransaction
      name ID #REQUIRED
      beginsWhen CDATA #IMPLIED
      endsWhen CDATA #IMPLIED
      isSecureTransportRequired (no | persistent |
                                 transient) "no"
      isGuaranteedDeliveryRequired (true | false) false
      requires CDATA #IMPLIED
      resultsIn 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

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.
beginsWhen	A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to commence.	Optional Input..
endsWhen	A description of an event external to this	Optional Input.

	collaboration/transaction that normally causes this collaboration/transaction to conclude.	
isSecureTransportRequired	Any document exchanged as part of this transaction must be sent in manner that guarantees that it isConfidential, isTamperProof, and isAuthenticated	**no** Valid Values: {no, persistent, transient }
isGuaranteedDeliveryRequired	Delivery of every document exchanged as part of this transaction must be guaranteed	**false** Valid Values: {true, false}
requires	A description of a state external to this collaboration/transaction that is required before this collaboration/transaction to conclude.	Optional Input.
resultsIn	A description of a state that does not exist before the execution of this collaboration/transaction but will exists as a result of the execution of this collaboration/transaction .	Optional Input.

e. Business Transaction Activity

Element Name: BusinessTransactionActivity

Content Model:

```
<!ELEMENT BusinessTransactionActivity (Documentation*)>
<!ATTLIST BusinessTransactionActivity
          name CDATA #REQUIRED
          businessTransaction CDATA #REQUIRED
          fromAuthorizedRole CDATA #REQUIRED
          toAuthorizedRole CDATA #REQUIRED
          isConcurrent (true | false) "false"
          isLegallyBinding (true | false) "true"
          isSynchronous (true | false) "false"
          timeToPerform CDATA #IMPLIED>
```

Definition:

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. The business transaction activity can be executed more than once if the ***isConcurrent*** property is ***true***.

"fromAuthorizedRole" and "toAuthorizedRole" must match one of the AuthorizedRoles in the binary collaboration. "from AuthorizedRole" will become the requestor in the transaction, “toAuthorizedRole” will become the responder.

Parents:

- BinaryCollaboration

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.
businessTransaction	A reference, by name to the Business Transaction used by this Business Transaction Activity	Required Input.
fromAuthorizedRole	The name of the initiating role in Business Transaction Activity	Required Input.
toAuthorizedRole	Name of the responding role in Business Transaction Activity	Required Input.
timeToPerform	The time a responding role has to acknowledge business acceptance of a business document	Optional Input.
isLegallyBinding	Defines whether the business transaction as performed in this Business Transaction Activity is legally binding.	**true** Valid Values: {true, false}
isConcurrent	Defines whether more than one business transaction can be open during the same time period.	**false** Valid Values: {true, false}
isSynchronous	The Business Transaction is intended to be performed by this activity in a synchronous manner	**false** Valid Values: {true, false}

f. Collaboration Activity

Element Name: CollaborationActivity

DTD Declaration:

```
<!ELEMENT CollaborationActivity (Documentation*)>
<!ATTLIST CollaborationActivity
      name ID #REQUIRED
      fromAuthorizedRole CDATA #REQUIRED
      toAuthorizedRole CDATA #REQUIRED
      binaryCollaboration CDATA #REQUIRED>
```

Definition:

Defines the use of one binary collaboration within another.

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

"fromAuthorizedRole" and "toAuthorizedRole" must match one of the AuthorizedRoles in the binary collaboration. "from AuthorizedRole" will become the initiator in the used binary collaboration, “toAuthorizedRole” will become the responder.

Parents:

- BinaryCollaboration

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.
fromAuthorizedRole	The name of the initiating role in Collaboration Activity	Required Input
toAuthorizedRole	Name of the responding role in Collaboration Activity	Required Input.
binaryCollaboration	A reference, by name, to the Binary Collaboration used by this Collaboration Activity	Required 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
- DocumentFlow
- DocumentType
- EbXmlProcessSpecification
- MultiPartyCollaboration
- Package
- Performs
- RequestingBusinessActivity
- RespondingBusinessActivity
- Schema
- Transition

Attributes:

Attribute Name	Definition	Default Value
uri	Defines the URI (Uniform Resource Identifier) where external documentation is located.	No Default Value. Valid URI is required.

h. Document Flow

Element Name: DocumentFlow

Content Model:

```
<!ELEMENT DocumentFlow (Documentation*, DocumentType,
                        Attachment*)>
<!ATTLIST DocumentFlow
      isSuccess CDATA #REQUIRED
      isAuthenticated (true | false) "false"
      isConfidential (true | false) "false"
      isTamperProof (true | false) "false">
```

Definition:

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

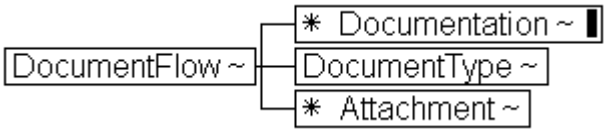
Parents:

- RequestingBusinessActivity
- RespondingBusinessActivity

Attributes:

Attribute Name	Definition	Default Value
isSuccess	An expression whose evaluation results in TRUE or FALSE as the determination of whether this DocumentFlow should be considered the successful initiation or conclusion of a BusinessTransaction.	Required Input.
isAuthenticated	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.	**false** Valid Values: {true, false}
isConfidential	The information entity is encrypted so that unauthorized parties cannot view the information	**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.	**false** Valid Values: {true, false}

Hierarchical View:



i. DocumentType

Element Name: DocumentType

DTD Declaration:

```
<!ELEMENT DocumentType  (Documentation*) >
<!ATTLIST DocumentType
                name  CDATA    #REQUIRED  >
```

Definition:

DocumentType is a generic name of a document. The definition of the document can be found in the associated Schema. Associated with the document can optionally be a set of attachments

Specifies the document type in the form of “SchemaName/DocumentType”

Parents:

- Attachment
- Schema

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

j. ebXmlProcessSpecification

Element Name: ebXmlProcessSpecification

DTD Declaration:

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

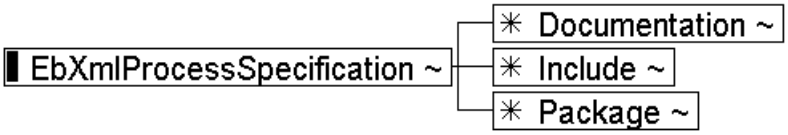
Definition:

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

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
uuid	Universally unique identifier.	No Default Value
version	Version of the specification.	No Default Value

Hierarchical Model:



k. Failure

Element Name: Failure

DTD Declaration:

```
<!ELEMENT Failure EMPTY>
<!ATTLIST Failure
      fromBusinessState CDATA #REQUIRED
      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 prior activity.	Required Input.
guardCondition	A reference to the status of the previous transaction. A fixed value of success, businessFailure, technicalFailure, or anyFailure	Optional Input. Valid Values: {Success, BusinessFailure, TechnicalFailure , AnyFailure}
guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition to success should happen or not. The expression can refer to the name or content of the document in the most recent DocumentFlow	Optional Input

l. Include

Element Name: Include

DTD Declaration:

```
<!ELEMENT Include   EMPTY  >
<!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:

- EbXmlProcessSpecification

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
uri	`Uniform Resource Indicator.`	No Default Value
uuid	`Universally unique identifier.`	No Default Value
version	Version of the included specification.	No Default Value

m. MultiParty Collaboration

Element Name: MultiPartyCollaboration

DTD Declaration:

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

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.

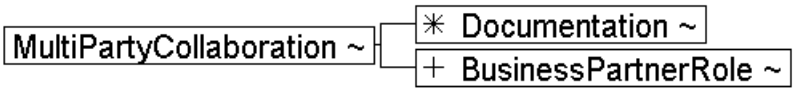
Parents:

- Package

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

Hierarchical Model:



n. Package

Element Name: Package

DTD Declaration:

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

Definition:

Defines a hierarchical name scope containing reusable elements.

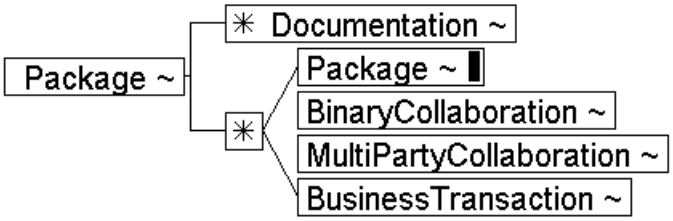
Parents:

- ebXmlProcessSpecification
- package

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

Hierarchical Model:



o. Performs

Element Name: Performs

DTD Declaration:

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

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	Name of the Authorized that will be performed by the business partner role, qualified with the name of the binary collaboration	Required Input

p. Requesting Business Activity

Element Name: RequestingBusinessActivity

DTD Declaration:

```
<!ELEMENT RequestingBusinessActivity  (Documentation*,
                                       DocumentFlow) >
<!ATTLIST RequestingBusinessActivity
    name                            CDATA          #REQUIRED
    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 activity that is performed by a role requesting commerce from another role. It specifies the Document Flow which will carry the request.

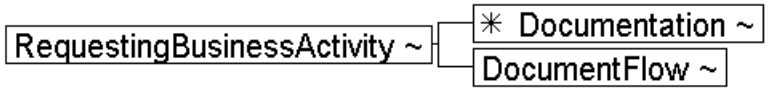
Parents:

- BusinessTransaction

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
isAuthorizationRequired	Must validate identity of originator against a list of authorized originators	**false** Valid Values: {true, false}
isIntelligibleCheckRequired	Receiving partner role must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt	**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.	**false** Valid Values: {true, false}
isNonRepudiationRequired	Requires the sending parties to save copies of the transacted documents before sending	**false** Valid Values: {true, false}
timeToAcknowledgeAcceptance	The time a responding role has to non-substantively acknowledge business acceptance of a business document.	No default value.
timeToAcknowledgeReceipt	The time a responding role has to acknowledge receipt of a business document	No default value.

Hierarchical Model:



q. Responding Business Activity

Element Name: RespondingBusinessActivity

DTD Declaration:

```
<!ELEMENT RespondingBusinessActivity  (Documentation*,
                                       DocumentFlow) >
<!ATTLIST RespondingBusinessActivity
    name                            CDATA          #REQUIRED
    isAuthorizationRequired         (true | false)  "false"
    isIntelligibleCheckRequired     (true | false)  "false"
    isNonRepudiationReceiptRequired (true | false) "false"
    isNonRepudiationRequired        (true | false) "false"
    timeToAcknowledgeAcceptance      CDATA          #IMPLIED
    timeToAcknowledgeReceipt         CDATA         #IMPLIED>
```

Definition:

A RespondingBusinessActivity is a business activity that is performed by a role responding to another business role's request for commerce.

It references the Document Flow which will carry the request.

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

Parents:

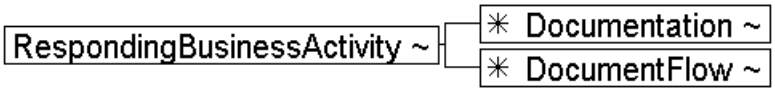
- BusinessTransaction

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
isAuthorizationRequired	Must validate identity of originator against a list of authorized originators	**false** Valid Values: {true, false}
isIntelligibleCheckRequired	Receiving partner role must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of	**false** Valid Values: {true, false}

	receipt	
isNonRepudiationReceiptRequired	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt.	**false** Valid Values: {true, false}
isNonRepudiationRequired	Requires the sending parties to save copies of the transacted documents before sending	**false** Valid Values: {true, false}
timeToAcknowledgeAcceptance	The time a responding role has to non-substantively acknowledge business acceptance of a business document.	No default value.
timeToAcknowledgeReceipt	The time a responding role has to acknowledge receipt of a business document	No default value.

Hierarchical Model:



r. Schema

Element Name:

DTD Declaration:

```
<!ELEMENT Schema (Documentation*, DocumentType*)>
<!ATTLIST Schema
      name CDATA #REQUIRED
      location CDATA #IMPLIED
      logicalModel CDATA #IMPLIED>
```

Definition:

A Schema is a collection of Document Definitions. The Schema 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 Schema. Typically this would be an ebXML core component context model.

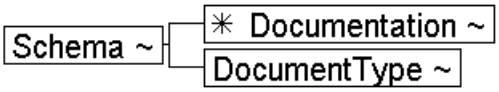
Parents:

- Package

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
location	Reference to an external source of the schema definition	No default value.
logicalModel	Reference is to where the logical model is	No default value.

Attributes:



s. Fork

Element Name: Fork

DTD Declaration:

```
<!ELEMENT Fork   EMPTY  >
<!ATTLIST Fork
          name  CDATA    #REQUIRED  >
```

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 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

t. Start

Element Name: Start

DTD Declaration:

```
<!ELEMENT Start   EMPTY  >
<!ATTLIST Start
          toBusinessState  CDATA    #REQUIRED  >
```

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	Name of an activity that is an allowed entry point for collaboration.	Required Input

u. Success

Element Name: Success

DTD Declaration:

```
<!ELEMENT Success EMPTY>
<!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 prior activity.	Required Input.
guardCondition	A reference to the status of the previous transaction. A fixed value of success, businessFailure, technicalFailure, or anyFailure	Optional Input. Valid Values: {Success, BusinessFailure, TechnicalFailure , AnyFailure}

guardExpression	An expression whose evaluation results in TRUE or FALSE to determine if this transition to success should happen or not. The expression can refer to the name or content of the document in the most recent DocumentFlow	Optional Input

v. Join State

Element Name: Join

DTD Declaration:

```
<!ELEMENT Join   EMPTY  >
<!ATTLIST Join
          name  CDATA    #REQUIRED  >
```

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 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
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.	**false** Valid Values: {true, false}

w. Transition

ELEMENT Name: Transition

DTD Declaration:

```
<!ELEMENT Transition (Documentation*)>
<!ATTLIST Transition
      onInitiation (true | false) "false"
      fromBusinessState CDATA #IMPLIED
      toBusinessState CDATA #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	Is used to specify this is a nested BusinessTransactionActivity and a second transaction is performed before returning to the this 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.
toBusinessState	The name of the state transitioned to	No default value.
guardCondition	A reference to the status of the previous transaction. A fixed value of success, businessFailure, technicalFailure, or anyFailure	Optional Input. 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 document in the most recent DocumentFlow	Optional Input

8.3 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.

2679

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
DocumentFlow	DocumentFlow
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
- TerminalState
- 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:

- DocumentType (attribute of DocumentFlow and of Schema)

8.4 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. EbXML specification element names are all contained within a name scope, usually a package. The set of packages describes a hierarchical name space, much like a directory structure.

The name of the element defined by its “name” attribute is the “simple name” of the element. The simple name is sufficient to reference that element within the same name scope or any parent name scope. Simple names may only contain the characters: a—z, A..Z, 0..9, “_”.

A name scope that contains another named element is referred to as the “parent” of that named element and the contained element is the “child” of the parent scope.

The “current” name scope is the one from which the scoped name reference is made.

A simple name is “in scope”, that is can be resolved, if it is in the current name scope or a parent name scope.

To access elements in other name scopes the name reference must be qualified.

A name qualifier shall precede the simple name with a slash (“/”) character separating the two names. The qualifying name shall specify the simple name of the scope in which the simple name may be found. Since a package also has a name within a name scope, it can also be qualified. Thus qualified names my reference any depth in the namespace hierarchy.

The first simple name in a qualified name shall be the root. The current name space shall be searched for the root and, if found, shall resolve that part of the name. If the root is not found in the current name scope the parent scope shall be searched, recursively, until the root is found. If the scoped name starts with “/” the root namespace shall be the one defined by EbXmlProcessSpecification.

Examples of scoped names:

“Foo” (Simple name)

“Billing/invoice” (Name “invoice” found in “Billing” package)

“/accounting/billing/foo” (name “foo” found in “billing” package found in “accounting” package which is in the ebXmlProcessSpecification)

8.5 Sample XML document against above DTD

```
<?xml version="1.0"?>

<!DOCTYPE EbXmlProcessSpecification SYSTEM
"ebXmlSpecificationDTD099_1.dtd">

<EbXmlProcessSpecification name="GenericQuoteOrder" version="1.1" uuid="[1234-
5678-901234]">
	<Package name="Ordering">
```

```
<Documentation>
    The OrderBT business transaction specifies that an "Order"
    document will initiate the transaction and that an
    OrderConfirmation will be a successful reply and that an
    "OrderDenied" document will be a failure reply (failure indicating
    that a business commitment was not made)
</Documentation>
<Schema name="ebXML1.0" location="ebXML.org"
logicalModel="ebXML.org/coreComponents">
    <DocumentType name="OrderDT"/>
    <DocumentType name="OrderConfirmationDT"/>
    <DocumentType name="OrderDeniedDT"/>
    <DocumentType name="QuoteRequestDT"/>
    <DocumentType name="QuoteDT"/>
    <DocumentType name="ShippingNoticeDT"/>
    <DocumentType name="PaymentNoticeDT"/>
    <DocumentType name="WaybillDT"/>
    <DocumentType name="PickupReceiptDT"/>
    <DocumentType name="WaybillIncompleteDT"/>
    <DocumentType name="DeliveryReceiptDT"/>
</Schema>

<BusinessTransaction name="OrderBT">
    <RequestingBusinessActivity name="OrderBA"
    isNonRepudiationRequired="true">
        <DocumentFlow documentType="OrderDT"
        isSuccess="true"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name="OrderConfirmationBA"
    isNonRepudiationRequired="true">
        <DocumentFlow documentType="OrderConfirmationDT"
        isSuccess="true"/>
        <DocumentFlow documentType="OrderDeniedDT"
        isSuccess="false"/>
    </RespondingBusinessActivity>
</BusinessTransaction>
<BusinessTransaction name="QuoteBT">
    <RequestingBusinessActivity name="QuoteRequestBA">
        <DocumentFlow documentType="QuoteRequestDT"
        isSuccess="true"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name="QuoteBA">
        <DocumentFlow documentType="QuoteDT"
        isSuccess="true"/>
    </RespondingBusinessActivity>
```

```
</BusinessTransaction>
<BusinessTransaction name="ShippingNoticeBT"
isSecureTransportRequired="no">
    <RequestingBusinessActivity name="ShippingNoticeBA">
        <DocumentFlow documentType="ShippingNoticeDT"
        isSuccess="true"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name="ShippingNoticeBA">
    </RespondingBusinessActivity>
</BusinessTransaction>
<BusinessTransaction name="PaymentNoticeBT">
    <RequestingBusinessActivity name="PaymentNoticeBA">
        <DocumentFlow documentType="PaymentNoticeDT"
        isSuccess="true"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name="PaymentNoticeBA">
        </RespondingBusinessActivity>
</BusinessTransaction>
<BinaryCollaboration name="OrderCollaborationCO"
timeToPerform="P2D">
    <Documentation>
        The "OrderCollaboration" specifies that it will start with an
        "OrderBT" business-transaction-activity which indicates
        the buy will initiate it.  If the order is confirmed it will.  If
        the order is confirmed it will proceed to a "Shipping
        Notice" from the sell role and then to a "PaymentNotice"
        from the buy role. OrderDenied from the OrderBT activity
        will conclude the collaboration with a failure.
    </Documentation>
    <Documentation>
        timeToPerform = 2 days
    </Documentation>
    <AuthorizedRole name="buyer"/>
    <AuthorizedRole name="seller"/>
    <BusinessTransactionActivity name="OrderBTA"
    businessTransaction="OrderBT" fromAuthorizedRole="buyer"
    toAuthorizedRole="seller"/>
    <BusinessTransactionActivity name="ShippingNoticeBTA"
    businessTransaction="ShippingNoticeBT"
    fromAuthorizedRole="seller" toAuthorizedRole="buyer"/>
    <BusinessTransactionActivity name="PaymentNoticeBTA"
    businessTransaction="PaymentNoticeBT"
    fromAuthorizedRole="buyer" toAuthorizedRole="seller"/>
    <Documentation>
```

```
                fromBusinessState and toBusinessState hold the name of
                the BusinessTransactionActivity
            </Documentation>
            <Start toBusinessState="OrderBTA"/>
            <Transition fromBusinessState="OrderBTA"
            toBusinessState="ShippingNoticeBTA"/>
            <Transition fromBusinessState="ShippingNoticeBTA"
            toBusinessState="PaymentNoticeBTA"/>
            <Success fromBusinessState="PaymentNoticeBTA"
            guardCondition="Success"/>
            <Failure fromBusinessState="OrderBTA"
            guardCondition="BusinessFailure"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="QuoteOrderCollaborationCO">
            <Documentation>
                The "QuoteOrderCollaboration" starts with a QuoteBT
                business transaction and then proceeds to reuse the
                "OrderCollaboration".
            </Documentation>
            <AuthorizedRole name="buyer"/>
            <AuthorizedRole name="seller"/>
            <BusinessTransactionActivity name="QuoteBTA"
            businessTransaction="QuoteBT" fromAuthorizedRole="buyer"
            toAuthorizedRole="seller"/>
            <Documentation>
                Here we see that the name and type of an activity does not
                have to be the same as we use the OrderCollaboration as an
                activity
            </Documentation>
            <CollaborationActivity name="OrderItCA"
            binaryCollaboration="OrderCollaborationCO"
            fromAuthorizedRole="buyer" toAuthorizedRole="seller"/>
            <Start toBusinessState="QuoteBTA"/>
            <Transition fromBusinessState="QuoteBTA"
            toBusinessState="OrderItCA"/>
            <Success fromBusinessState="OrderItCA"
            guardCondition="Success"/>
            <Failure fromBusinessState="OrderItCA"
            guardCondition="BusinessFailure"/>
        </BinaryCollaboration>
        <MultiPartyCollaboration name="BuySellMC">
            <Documentation>
                In the BuySell multi-party collaboration we define the
                business partners "buyer" and "seller" which perform the
                "QuoteOrderCollaboration"
```

```
                </Documentation>
                <BusinessPartnerRole name="buyer">
                        <Performs
                        authorizedRole="QuoteOrderCollaborationCO/buyer"/>
                </BusinessPartnerRole>
                <BusinessPartnerRole name="seller">
                        <Performs
                        authorizedRole="QuoteOrderCollaborationCO/seller"/>
                </BusinessPartnerRole>
        </MultiPartyCollaboration>
</Package>
<!-- End of Ordering-->
<Package name="Shipping">
        <Documentation>
                A sender requests shipping from a carrier
        </Documentation>
        <BusinessTransaction name="ShippingBT">
                <RequestingBusinessActivity name="WaybillBA"
                isIntelligibleCheckRequired="true">
                        <DocumentFlow documentType="WaybillDT"
                        isSuccess="true"/>
                </RequestingBusinessActivity>
                <RespondingBusinessActivity name="PickupReceiptBA">
                        <DocumentFlow documentType="PickupReceiptDT"
                        isSuccess="true"/>
                        <DocumentFlow documentType="WaybillIncompleteDT"
                        isSuccess="false"/>
                </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="DeliveryAcknowledgementBT">
                <RequestingBusinessActivity name="DeliveryReceiptBA">
                        <DocumentFlow documentType="DeliveryReceiptDT"
                        isSuccess="true"/>
                </RequestingBusinessActivity>
                <RespondingBusinessActivity name="DeliveryReceiptBA">
                </RespondingBusinessActivity>
        </BusinessTransaction>
        <BinaryCollaboration name="ShipCollaborationCO">
                <AuthorizedRole name="sender"/>
                <AuthorizedRole name="shipper"/>
                <BusinessTransactionActivity name="ShippingBTA"
                businessTransaction="ShippingBT" fromAuthorizedRole="sender"
                toAuthorizedRole="shipper"/>
                <BusinessTransactionActivity
                name="DeliveryAcknowledgementBTA"
```

```
                    businessTransaction="DeliveryAcknowledgementBT"
                    fromAuthorizedRole="shipper" toAuthorizedRole="sender"/>
                    <Start toBusinessState="ShippingBTA"/>
                    <Transition fromBusinessState="ShippingBTA"
                    toBusinessState="DeliveryAcknowledgementBTA"/>
                    <Success fromBusinessState="DeliveryAcknowledgementBTA"
                    guardCondition="Success"/>
                    <Failure fromBusinessState="ShippingBTA"
                    guardCondition="BusinessFailure"/>
            </BinaryCollaboration>
            <MultiPartyCollaboration name="BuySellShip">
                    <Documentation>
                            <!-- CDATA is used as example because of the & character
                            in the text -->
                            <![CDATA[
                                    Multiparty business collaboration : Synthesizing
                                    two service interactions BuySell & Ship across
                                    three kinds of business partners "buyer" "seller" and
                                    "carrier".  Note that the "seller" performs two roles
                                    within this collaboration.
                            ]]>
                    </Documentation>
                    <BusinessPartnerRole name="buyer">
                            <Performs
                            authorizedRole="OrderCollaborationCO/buyer"/>
                    </BusinessPartnerRole>
                    <BusinessPartnerRole name="seller">
                            <Performs
                            authorizedRole="OrderCollaborationCO/seller"/>
                            <Performs
                            authorizedRole="ShipCollaborationCO/sender"/>
                            <Transition
                            fromBusinessState="OrderCollaborationCO/ShippingNotic
                            eBTA"
                            toBusinessState="ShipCollaborationCO/ShippingBTA"/>
                    </BusinessPartnerRole>
                    <BusinessPartnerRole name="carrier">
                            <Performs authorizedRole="shipper"/>
                    </BusinessPartnerRole>
            </MultiPartyCollaboration>
    </Package>
</EbXmlProcessSpecification>
```

9 Common Modeling Elements

9.1 Data typing

XML currently has limited data typing capability for attributes and no data typing capability for elements. EbXML recognizes that data typing is a requirement for business transactions using XML. The World Wide Web Consortium (W3C) has defined a group of core data types as part of the XML Schema Specification (XML Schema Part 2: Data types, W3C Candidate Recommendation, 24 October 2000). The data types defined by the W3C will be used for global data types.

9.1.1 Global Data types

The following two charts defines the global data types that will be used for the *ebXML Business Process Specification Schema.* The first chart defines the data types that are currently available for use natively within XML DTDs. The second chart defines the proposed data types available for use with W3C XML Schema Specification.

Datatypes Natively Available in DTDs	
Datatype	**Definition**
ID	ID represents the ID attribute type from [XML 1.0 Recommendation (Second Edition)].
IDREF	IDREF represents the IDREF attribute type from [XML 1.0 Recommendation (Second Edition)].
IDREFS	IDREFS represents the IDREFS attribute type from [XML 1.0 Recommendation (Second Edition)].
CDATA	CDATA (Character Data) represents white space normalized strings.
ENTITY	ENTITY represents the ENTITY attribute type from [XML 1.0 Recommendation (Second Edition)].
ENTITIES	ENTITIES represents the ENTITIES attribute type from [XML 1.0 Recommendation (Second Edition)].
NMTOKEN	NMTOKEN represents the NMTOKEN attribute type from [XML 1.0 Recommendation (Second Edition)].
NMTOKENS	NMTOKENS represents the NMTOKENS attribute type from [XML 1.0 Recommendation (Second Edition)].
NOTATION	NOTATION represents the NOTATION attribute type from [XML 1.0 Recommendation (Second Edition)].

The table below shows the datatypes that are not natively provided in DTDs. These datatypes will be available in W3C Schema Specification, as well as the Regular Language description for XML (RELAX) schema language that has recently been submitted to ISO.

Although the datatypes are not currently natively available in DTDs (native XML parsers cannot validate) processes can be used to validate the datatypes external from the XML parser.

Datatypes Not Available in DTDs	
Datatype	**Definition**
string	The string datatype represents character strings in XML.
boolean	The boolean datatype has the value space required to support the mathematical concept of binary-valued logic: {true, false}.
float	The float datatype corresponds to the IEEE single-precision 32-bit floating point type [IEEE 754-1985].
double	The double datatype corresponds to IEEE double-precision 64-bit floating point type [IEEE 754-1985].
decimal	The decimal datatype represents arbitrary precision decimal numbers.
timeDuration	The timeDuration datatype represents a duration of time.
recurringDuration	The recurringDuration datatype represents a specific period of time that recurs with a specific frequency, starting from a specific point in time.
binary	The binary datatype represents arbitrary binary data.
uriReference	The uriReference datatype represents a Uniform Resource Indentifier (URI) Reference.
Qname	QName represents XML qualified names.
token	The token datatype represents tokenized strings.
language	The language datatype represents natural language identifiers as defined by [RFC 1766].
Name	Name represents XML Names.
NCName	NCName represents XML "non-colonized" Names.
integer	The integer datatype is derived from decimal by fixing the value of scale to be 0.
nonPositiveInteger	nonPositiveInteger is derived from integer by setting the value of maxInclusive to be 0.
negativeInteger	The negativeInteger datatype is derived from nonPositiveInteger by setting the value of maxInclusive to be -1.
long	the long datatype is derived from integer by setting the value of maxInclusive to be 9223372036854775807 and minInclusive to be -9223372036854775808. The base type of long is integer.
int	The int datatype is derived from long by setting the value of maxInclusive to be 2147483647 and minInclusive to be -2147483648. The base type of int is long.

short	short is derived from int by setting the value of maxInclusive to be 32767 and minInclusive to be -32768. The base type of short is int.
byte	The byte datatype is derived from short by setting the value of maxInclusive to be 127 and minInclusive to be -128. The base type of byte is short.
nonNegativeIneger	The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to be 0.
unsignedLong	The unsignedLong datatype is derived from nonNegativeInteger by setting the value of maxInclusive to be 18446744073709551615. The base type of unsignedLong is nonNegativeInteger.
unsignedInt	The unsignedInt datatype is derived from unsignedLong by setting the value of maxInclusive to be 4294967295. The base type of unsignedInt is unsignedLong.
unsignedShort	The unsignedShort datatype is derived from unsignedInt by setting the value of maxInclusive to be 65535. The base type of unsignedShort is unsignedInt.
unsignedByte	The unsignedByte datatype is derived from unsignedShort by setting the value of maxInclusive to be 255. The base type of unsignedByte is unsignedShort.
positiveInteger	The positiveInteger datatype is derived from nonNegativeInteger by setting the value of minInclusive to be 1.
timeInstant	The timeInstant datatype represents a specific instant of time.
time	The time datatype represents an instant of time that recurs every day.
timePeriod	The timePeriod datatype represents a specific period of time with a give start and end.
date	The date datatype represents a timePeriod that starts at midnight of a specified day and lasts until midnight the following day.
month	The month datatype represents a timePeriod that starts at midnight on the first day of the month and lasts until the midnight that ends the last day of the month.
year	The year datatype represents a timePeriod that starts at the midnight that starts the first day of the year and ends at the midnight that ends the last day of the year.
century	The century datatype represents a timePeriod that starts at the midnight that starts the first day of the century and ends at the midnight that ends that last day of the century.
recurringDate	The recurringDate datatype is a date that recurs, specifically a day of the year such as the third of May.
recurringDay	The recurringDay datatype is a day that recurs, specifically a day of the month such as the 5th of the month.

9.1.2 Local Datatypes

Local datatypes used within ebXML, i.e., currency, will be developed by the ebXML Core Components Working Group.

9.2 Business signal structures

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. Here are the DTD's for receiptAcknowledgment (from the RosettaNet website, courtesy of RosettaNet, and Edifecs).[4] and for acceptanceAcknowledgement and exception.

9.2.1 ReceiptAcknowledgment DTD

```
<!--

   RosettaNet XML Message Schema
   AcknowledgmentOfReceipt_MS_D02_00.dtd (31-Oct-2000
08:26)
   This document has been prepared by Edifecs
(http://www.edifecs.com/)
   based On the Business Collaboration Framework from
requirements
   in conformance with the RosettaNet methodology.

-->


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

<!ELEMENT ReceiptAcknowledgment (
          NonRepudiationInformation? ) >

<!ELEMENT NonRepudiationInformation (
          OriginalMessageDigest ) >

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

9.2.2 AcceptanceAcknowledgement DTD

[4] From RosettaNet Implementation Framework: Core Specification, Version: Release 2.00.00, 3 January 2001

```
<!--
   ebXML XML Message Schema
   AcceptanceAcknowledgment_v099.dtd (19-Mar-2001 20:26)
This document is a derivation of the RosettaNet
ReceiptAcknowledgement signal, used by permission.
-->

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

<!ELEMENT AcceptanceAcknowledgment (
          NonRepudiationInformation? ) >

<!ELEMENT NonRepudiationInformation (
          OriginalMessageDigest ) >

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

9.2.3 Exception Signal DTD

```
<!--

   ebXML XML Message Schema
   ebxmlException_v099.dtd (19-Mar-2001 18:25)
   This document is a derivation of the RosettaNet
Exception signal, used by permission
-->

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

<!ELEMENT Exception (
          ExceptionDescription ,
          GlobalExceptionTypeCode ) >

<!ELEMENT ExceptionDescription (
          errorClassification ,
          errorDescription ,
          offendingMessageComponent? ,
          OffendingMessageContent? ) >

<!ELEMENT errorClassification
           ( GlobalMessageExceptionCode ) >

<!ELEMENT GlobalMessageExceptionCode
           ( #PCDATA ) >

<!ELEMENT errorDescription
           ( FreeFormText ) >

<!ELEMENT FreeFormText
           ( #PCDATA ) >
<!ATTLIST FreeFormText
```

```
                        xml:lang CDATA #IMPLIED >

            <!ELEMENT offendingMessageComponent
                        ( GlobalMessageComponentCode ) >

            <!ELEMENT GlobalMessageComponentCode
                        ( #PCDATA ) >

            <!ELEMENT OffendingMessageContent (
                      XmlContentAddressingStandard ,
                      OffendingMessageElement* ) >

            <!ELEMENT XmlContentAddressingStandard (
                      standardName ,
                      standardVersion? ) >

            <!ELEMENT standardName
                        ( GlobalAdministeringAuthorityCode ) >

            <!ELEMENT GlobalAdministeringAuthorityCode
                        ( #PCDATA ) >

            <!ELEMENT standardVersion
                        ( VersionIdentifier ) >

            <!ELEMENT VersionIdentifier
                        ( #PCDATA ) >

            <!ELEMENT OffendingMessageElement (
                      xmlContentAddress ,
                      errorDescription? ,
                      ValidationConstraint? ) >

            <!ELEMENT xmlContentAddress
                        ( FreeFormText ) >

            <!ELEMENT ValidationConstraint
                        ( #PCDATA ) >

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

10 Production Rules

The Specification Production rules provide the prescriptive definition necessary to translate a UML Specification Model into an XML Specification Document and the well-formed rules necessary to populate an XML Specification Document.

The Specification Production rules provide the prescriptive definition necessary to translate a UML Specification Model into a XML Specification Document. 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) that has no corresponding definition is rendered as a string. Time is formatted following ISO 8601 CCYYMMDDThhmmss.sssZ.
9. Each class is given an optional “Documentation*” element which is intended for annotation of the specification instances. This is not modeled.

11 References

1. UN/CEFACT Modelling Methodology (CEFACT/TMWG/N090R8E)
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

Copyright Statement