UN/CEFACT – ebXML Business Process Specification Schema

25 August 2003
Version 1.09

Review Copy

For UN/CEFACT TMG Distribution and Use Only

Not Released For Use or Implementation by Other Parties!!!

1 Status of This Document

This *UN/CEFACT –Technical Specification* is being developed in accordance with the UN/CEFACT/TRADE/22 Open Development Process for Technical Specifications. It has been approved by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) Techniques and Methodology Group (TMG) for internal TMG review as defined in the Open Development Process.

Distribution of this document is limited to the TMG.

This version: *UN/CEFACT – ebXML Business Process Specification Schema*, Version 1.09 of 25 August 2003

Previous Draft (For Review) version: *UN/CEFACT – ebXML Business Process Specification Schema* Version 1.05 of 15 July 2002

Current "Approved" (For Implemenatation) version: ebXML Business Process Specification Schema Version 1.01 of 11 May 2001

2 Table of Contents

1 Status of This Document iii
2 Table of Contents iv
3 Introduction vii
3.1 Summary of Contents of Document ix
3.2 Audience ix
3.3 Related Documents ix
3.4 Prerequisites ix
4 Design Objectives 1
4.1 Goals/Objectives/Requirements/Problem Description 1
4.2 Caveats and Assumptions 1
4.2.1 Relationship between *ebXML Business Process Specification Schema* and UMM 1
5 Language Overview 5
5.1 UML Representation of Business Process Specification Schema 7
5.2 XML Schema representation of Business Process Specification Schema 7
5.3 UMM Business Process Interaction Patterns 7
5.4 Business Signal Definitions 7
5.5 Production Rules 8
5.6 Relationship to CPP/CPA 8
5.7 Relationship to Business Documents 8
5.8 Relationship to ebXML Message Service Specification 8
5.9 Key Concepts of the ebXML Business Process Specification Schema 9
5.10 How to use the ebXML Business Process Specification Schema 13
5.11 How ebXML Business Process Specification Schema is used with other ebXML specifications 13
5.12 How to design collaborations and transactions, re-using at design time 15
5.12.1 Packages and Includes 17
5.12.2 Substitution Sets 18
5.12.3 Specify a Business Transaction and its Business Document Flow 19
5.12.4 Specify a Binary Collaboration 29
5.12.5 Specify a MultiParty Collaboration 33
5.12.6 Specify a Choreography 36
5.12.7 The whole model 42
5.13 Core Business Transaction Semantics 43
5.13.1 Interaction Predictability 43
5.13.2 Creating legally binding contracts 46
5.13.3 Non-Repudiation 47

5.13.4 Authorization security 48
5.13.5 Document security 48
5.13.6 Reliability 49
5.13.7 Parameters required for CPP/CPA 49

5.14 Run time Business Transaction semantics 49
5.14.1 Timeouts 50
5.14.2 Protocol Exceptions 51
5.14.3 Computation of the status of a Business Transaction Activity 55

5.15 Runtime Collaboration Semantics 56

5.16 Where the ebXML Business Process Specification Schema May Be Implemented 56

5.17 Guidelines for Business Service Interface Interoperability 57

5.18 Collaboration and transaction well-formedness rules 57

6 ebXML Business Process Specification Schema – 59

6.1 Documentation for the Schema 59
6.1.1 element Attachment 59
6.1.2 element AttributeSubstitution 61
6.1.3 element BinaryCollaboration 62
6.1.4 element BinaryCollaboration/Role 63
6.1.5 element BusinessDocument 63
6.1.6 element BusinessPartnerRole 65
6.1.7 element BusinessTransaction 66
6.1.8 element BusinessTransactionActivity 67
6.1.9 element CollaborationActivity 69
6.1.10 element ConditionExpression 69
6.1.11 element Decision 70
6.1.12 element Documentation 70
6.1.13 element DocumentEnvelope 72
6.1.14 element DocumentSubstitution 72
6.1.15 element Failure 74
6.1.16 element Fork 75
6.1.17 element Include 76
6.1.18 element Join 76
6.1.19 element MultiPartyCollaboration 77
6.1.20 element Namespace 78
6.1.21 element Namespaces 78
6.1.22 element Package 79
6.1.23 element Performs 79
6.1.24 element ProcessSpecification 80
6.1.25 element RequestingBusinessActivity 82
6.1.26 element RespondingBusinessActivity 83
6.1.27 element Start 83
6.1.28 element SubstitutionSet 85
6.1.29 element Success 86
6.1.30 element Transition 86
6.1.31 simpleType GUID 87
6.1.32 simpleType GUIDREF 87

6.2 XML to UML cross-reference 89
6.3 Scoped Name Reference 90
6.4 Sample XML document against above Schema 91
7 Business signal structures 92
7.1.1 Signal Schema 92
7.1.2 ReceiptAcknowledgment Signal Schema 95
7.1.3 AcceptanceAcknowledgement Signal Schema 96
7.1.4 Exception Signal Schema 96
8 EDI support 98
9 Production Rules 98
Appendix A: Sample XML Business Process Specification Schema Instance 100
Appendix B: Sample XML Signals 102
10 References 105
11 Disclaimer 106
12 Contact Information 106
13 Copyright Statement 107

3 Introduction

Executive Summary

The ebXML Business Process Specification Schema technical specification defines a standard language by which business systems may be configured to support execution of business collaborations consisting of business transactions. It is based upon prior UN/CEFACT work, specifically the metamodel behind the UN/CEFACT Modeling Methodology (UMM) defined in the "UN/CEFACT Modeling Methodology - Meta Model - Revision 12 (2003-01-17)" specification.

The BPSS technical specification 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 are defined in the UMM specification. A pattern is not executable, it rather specifies the type of the message exchange (request, response and signals) that applies for a given business transaction definition. It is a way to define classes of business transaction definitions. These patterns could potentially be related to different classes of electronic commerce transactions.

The current version of the BPSS technical specification addresses collaborations between two parties (Binary Collaborations). Collaborations involving more than two business partners (Multiparty Collaborations) have been deprecated.

3.1 Summary of Contents of Document

This document describes the ebXML Business Process Specification Schema.

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

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

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

3.2 Audience

The primary audience is technical implementers of ebXML. We define a business process analyst as someone who applies the UN/CEFACT Modeling Methodology (UMM) which defines a process that centers around interviewing business people.

An additional audience are 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.

3.3 Related Documents

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

- ebXML Technical Architecture Specification, version 1.04
- UN/CEFACT Core Components Dictionary, version 1.04
- ebXML Naming Convention for Core Components, version 1.04
- ebXML Collaboration-Protocol Profile and Agreement Specification V2.0
- ebXML Business Process and Business Information Analysis Overview, version 1.0
- ebXML Business Process Analysis Worksheets & Guidelines, version 1.0
- ebXML E-Commerce Patterns, version 1.0
- ebXML Catalog of Common Business Processes, version 1.0
- ebXML Message Service Specification V2.0
- UN/CEFACT Modeling Methodology (UMM) as defined in the N090R12 specification

3.4 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 as defines in UN/CEFACT's Modeling Methodology (UMM)

- The UML syntax and semantics
- The Extensible Markup Language (XML)

4 Design Objectives

4.1 Goals/Objectives/Requirements/Problem Description

BPSS Instances describe interoperable business processes that allow business partners to collaborate. These models must be executed by software components that collaborate on behalf of the business partners.

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

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

A business process specification created with the ebXML Business Process Specification Schema is referred to as a BPSS instance.

The *ebXML Specification Schema* is available as an XML Schema (http://www.w3.org/2001/XMLSchema) format at this loacation: http:/webster.disa.org/cefact-groups/tmg/downloads/general/for_review/BPSS-V1pt09-DRAFT.xsd. A UML description of elements of the schema is found in relevant sections of this document.

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

The XML Schema provides the specification for XML based BPSS instances.

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

4.2 Caveats and Assumptions

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

It is recommended that the preferred methodology for creating an ebXML BPS shall be UN/CEFACT Modeling Methodology (UMM).

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

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

The UN/CEFACT Modeling Methodology (UMM) is a set of architectures, methodologies, business semantics, ontologies and reference models. The UMM offers a formal methodology for describing any Open-edi scenario as

defined in ISO/IEC 14662, Open-edi Reference Model. Examples of an Open-edi scenario are purchasing and inventory management. The primary scope of the UMM is to provide "a perspective of business transactions limited to those aspects regarding the making of business decisions and commitments among organizations, which are needed for the description of a business transaction". The UMM provides a procedure for specifying (modelling) business processes involving information exchange in a technology neutral, implementation-independent manner.

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

The UMM Meta Model is a description of business semantics that allows Trading Partners to capture the details for a specific business scenario (a Business Process) using a consistent modeling methodology. A Business Process specification 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 Entities, expressed in an appropriate format (XML, EDI, UBL, …). At a lower level, Business Processes can be composed of re-useable Common Business Processes, and Business Information Entities can be composed of re-useable Core Components. Common Business Processes and Business Information Entities reside in a UMM Business Library.

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

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

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

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

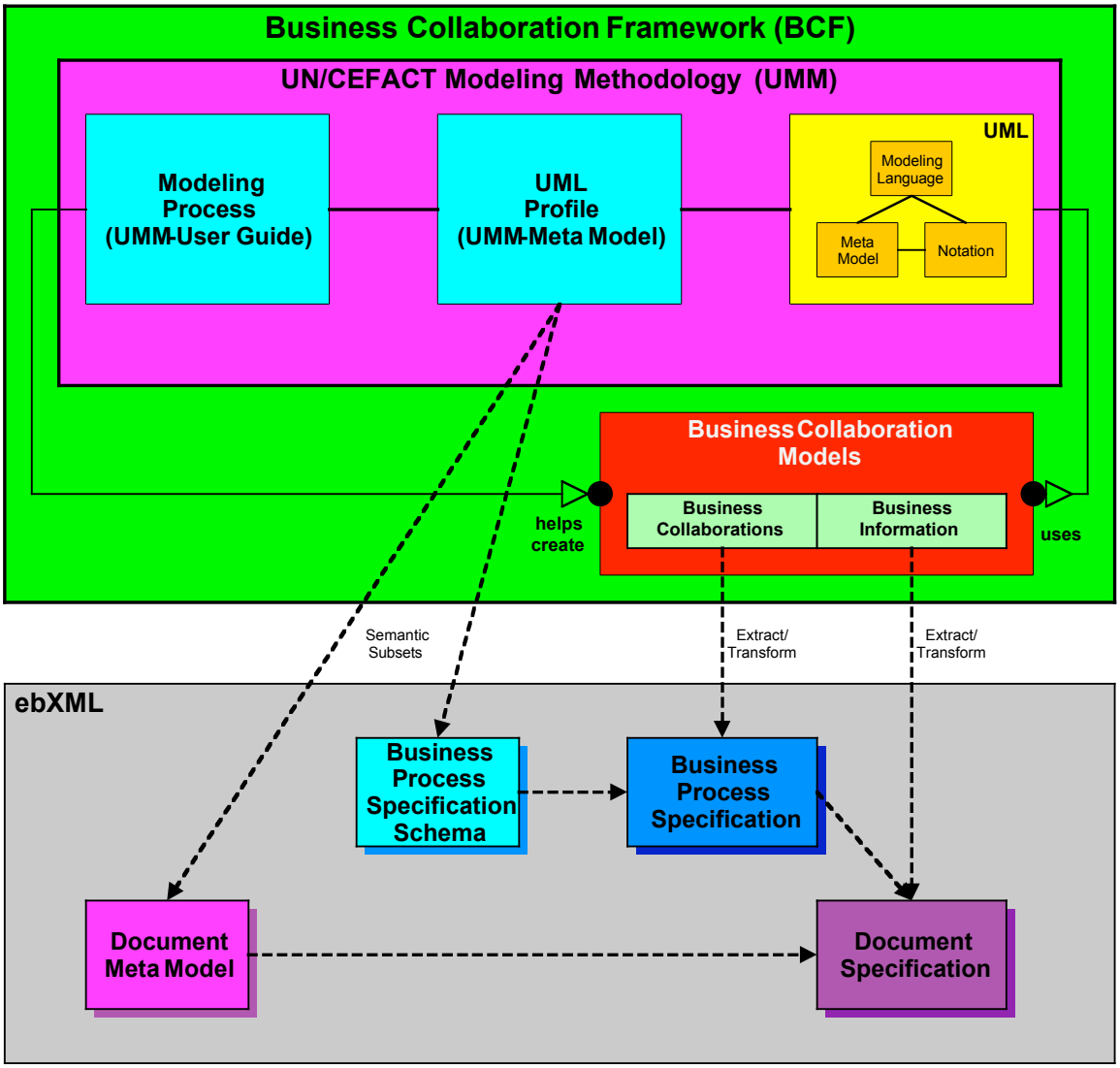


Figure 1. UMM Metamodel and *ebXML Business Process Specification Schema*

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

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

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

The UN/CEFACT UMM and CC Specification are not part of the formal set of ebXML specifications.

The remainder of this document focuses on the *ebXML Business Process Specification Schema* and Business Process Specifications created with it. It recommended that proper Business Process and Information Modeling using the UMM has taken place prior to beginning the activity of creating a Business Process Specification.

5 Language Overview

The ebXML *Business Process Specification Schema* defines a standard language for business process specification. As such, it works with the ebXML Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA) specifications to bridge the gap between Business Process Modeling and the configuration of ebXML compliant e-commerce software.

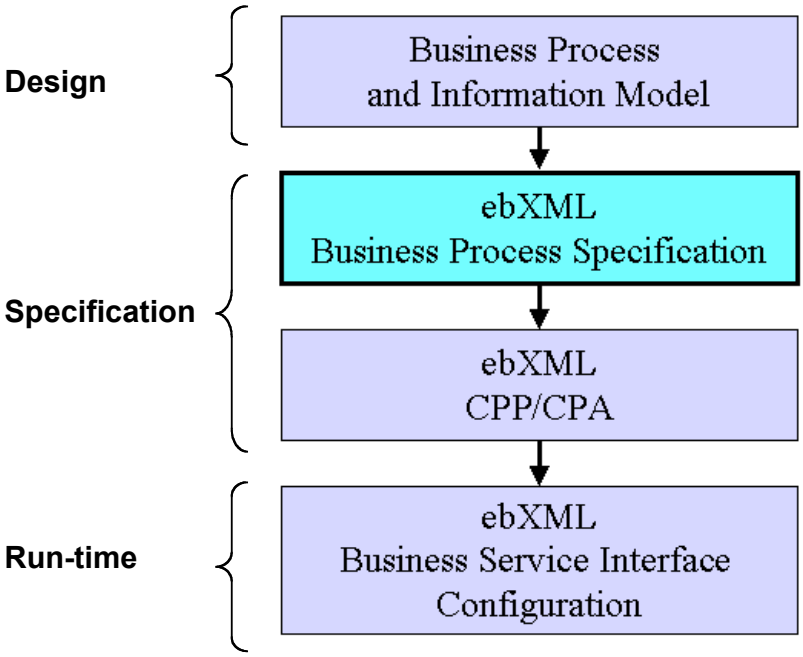


Figure 2: Business Process Specification and Business Service Interface Configuration

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

Based on this Business Process and Information Model and using the ebXML *Business Process Specification Schema* the user will then extract and format the nominal set of elements necessary to configure an ebXML runtime system in order to execute a set of ebXML business transactions. The result is a *BPSS instance*.

Alternatively the ebXML *BPSS instance* may be created directly, without prior explicit business process modeling.

A *BPSS instance* contains the specification of Business Transactions and the choreography of these Business Transactions into Business Collaborations.

This *BPSS instance* is then the input to the formation of ebXML trading partner Collaboration Protocol Profiles and Collaboration Protocol Agreements.

These ebXML trading partner Collaboration Protocol Profiles and Collaboration Protocol Agreements in turn serve as configuration files for

Business Service Interface (BSI) software component. The Business Service Interface Software represents any ebXML compliant component, which is able to be, configured from an ebXML BPSS instance and a CPA.

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

- UML representation of the *Business Process Specification Schema semantics*
- XML Schema definition of the *Business Process Specification Schema.* Each BPSS instance must conform to this schema definition.
- Production Rules defining the mapping from the UML representation of the *Business Process Specification Schema* to the XML Schema version
- Business Signal Definitions

Together these components allow you to specify the run time aspects of a business process model within the limitations of this current version of the BPSS. However, all the parameters of the ebXML *Business Process Specification Schema* are intented to be specified at design time. None of these parameters are specified or inferred at run-time.

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

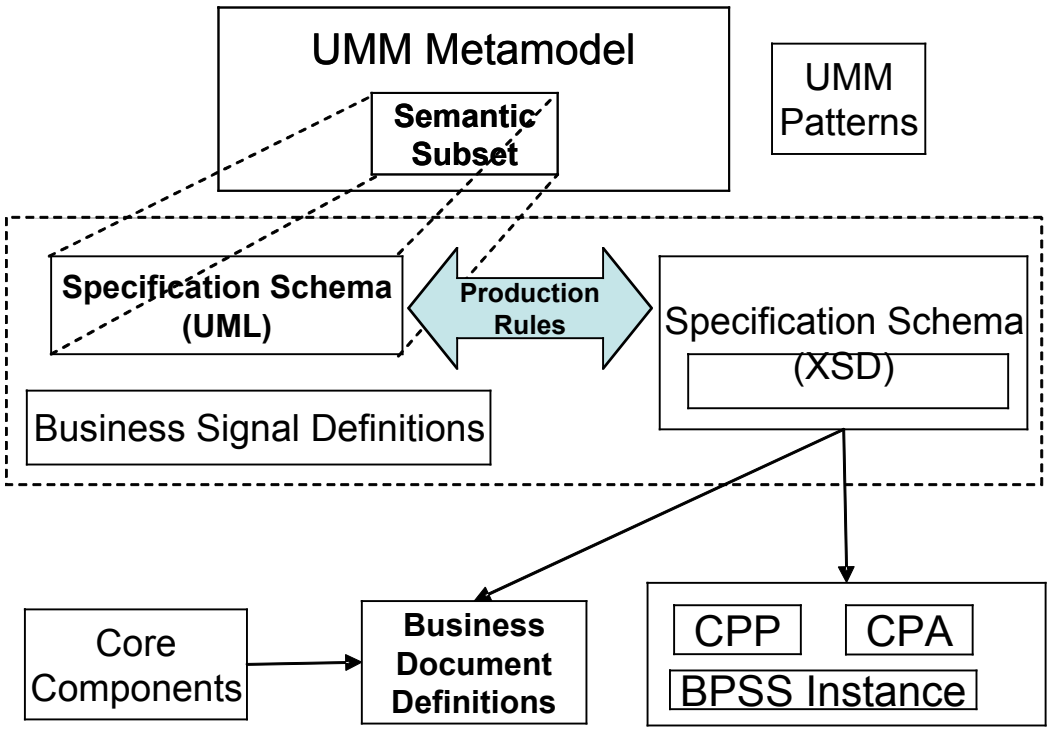


Figure 3: Relationship of *ebXML Business Process Specification Schema* to UMM, CPP/CPA and Core Components

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

5.1 UML Representation of Business Process Specification Schema

The UML representation of the *ebXML Business Process Specification Schema* is a semantic subset of the metamodel behind UMM as specified in UN/CEFACT Modeling Methodology - Meta Model - Revision 12 (2003-01-17). The UML representation of the ebXML *Business Process Specification Schema* is a UML Class Diagram.

5.2 XML Schema representation of Business Process Specification Schema

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

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

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

5.3 UMM Business Process Interaction Patterns

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

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

The Business Transaction Interaction Patterns set forth in the UN/CEFACT Modeling Methodology illustrate recommended permutations of message sequences as determined by the type of business transaction defined and the timing policies specified in the transactions. While the UMM patterns themselves are not part of the ebXML specifications, all the security and timing parameters required to express the pattern properties are provided as attributes of elements in the ebXML *Business Process Specification Schema*.

5.4 Business Signal Definitions

A business signal is an object that is transmitted back to an activity that initiated the transfer of execution control. Business signals have specific business purpose and are separate from lower protocol and transport signals as specified in the ebXML Message Service Specification. The state of a given business transaction activity instance can be explicitely calculated at run-time by evaluating these signals. As such they are instrumental in establishing a business collaboration protocol that garantees that the representation of the state of a business collaboration instance for each party,

is the strictly identical for both parties. This is what we reference as “state alignment”.

The structures of ebXML business signals are ‘universal’ and do not vary from transaction to transaction. Thus, they can be defined once and for all. These schemas are included in the ebXML *BPSS technical specification* itself.

The Business Process Specification provides both the choreography of business signals, and the structure definition of the business payload of a business signal. The ebXML Message Service Specification provides a reliable messaging infrastructure upon which the ebXML BPSS technical specification builts its protocol for business state alignment via the use of business signals. The business signal payload structures provided herein are optional and normative and are intended to provide business and legal semantics to the business signals.

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

5.5 Production Rules

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

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

5.6 Relationship to CPP/CPA

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

5.7 Relationship to Business Documents

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

5.8 Relationship to ebXML Message Service Specification

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

5.9 Key Concepts of the ebXML Business Process Specification Schema

The ebXML *Business Process Specification Schema* specifies the structure and semantics of machine processable business collaborations definitions. These semantics are aligned with the one of UMM and represent a subset of the UMM semantics.

At a high level a business collaboration consists of a set of roles collaborating through a set of choreographed transactions by exchanging business documents.

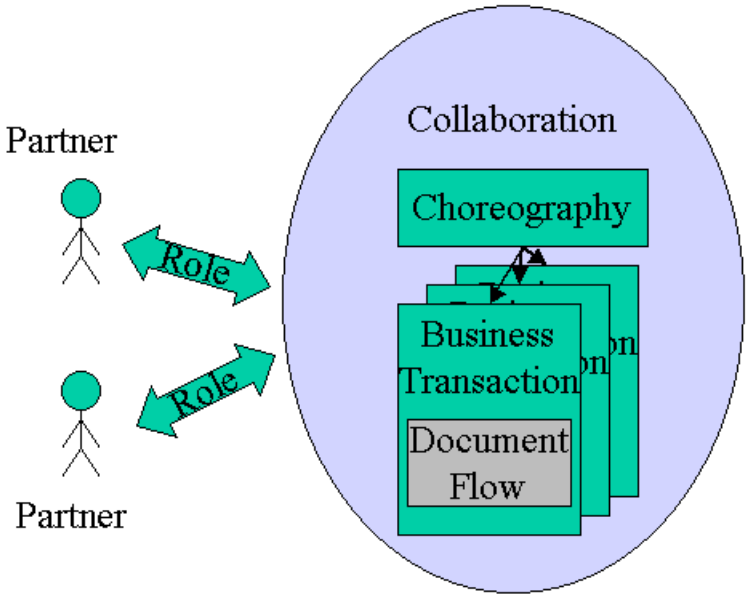


Figure 4. Illustration of the basic semantics of a business collaboration

Two or more business partners participate in the business collaboration through roles. The roles always exchange messages in the context of Business Transactions. Each Business Transaction consists of one or two predefined Business document flows. One or more Business Signals may additionally be exchanged as part of a Business Transaction to ensure state alignment of both parties. The business transactions are performed relative to each other as part of a choreography.

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

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

1. Business Collaborations

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

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

Binary Collaborations are between two roles only.

Multiparty Collaborations are between more than two roles, but such Multiparty Collaborations are always synthesized from two or more Binary Collaborations. For instance if Roles A, B, and C collaborate and all parties interact with each other, there will be a separate Binary Collaboration between A and B, one between B and C, and one between A and C. The Multiparty Collaboration will be the synthesis of these three Binary Collaborations. The concepts developed to specify multi-party collaboration are experimental and are being deprecated. It is recommended not to use this capability of the specification as it might change substantially in future releases. The implementation of this feature is therefore optional for any compliant ebXML Business Service Interface.

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

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

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

2. Business Transactions

A Business Transaction represents an atomic unit of work in a trading arrangement between two business partners. The scope of the BPSS technical specification is not to cover how BPSS Business Transactions are related to trading activities between business partners. This is the role of the UMM. A Business Transaction is conducted between two parties playing opposite roles in the transaction. The roles are always a requesting role and a responding role. They are not specific roles like buyer or seller. These roles will be specified at the Business Transaction Activity level, when the Business Transaction definition is used for a specific purpose.

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

Unlike a Binary Collaboration, however, the Business Transaction is atomic; it cannot be decomposed into lower level Business Transactions that could be reused independently of each other.

A Business Transaction is a very specialized and very constrained protocol, in order to achieve very precise and enforceable transaction semantics. These semantics are expected to be enforced by the software managing the transaction, i.e. an ebXML Business Service Interface (BSI) software component.

A Business Transaction will always either succeed or fail both from a protocol and a business perspective. If it succeeds from both perspectives it may be designated as legally binding between the two partners, or otherwise govern their collaborative activity. If it fails it is null and void, and each partner must relinquish any mutual claim established by the transaction. In addition, if it fails from protocol perspective, each party must synchronize their state to the state prior the start of the transaction. For instance, a purchase order state should advance to “sent” when and only when a protocol success is reported by the BSI. In case of a business failure, the state has already been “synchronized” and it is the duty of each application to take the proper actions. A Business failure is any failure that is identified by an application during the processing of the business document(s) and based on information not available to the BPSS. For instance, a “reject purchase order” response document would be considered as a business failure. In this case, it is the role of the applications to mark the state of the purchase order appropriately.

3. Business Document flows

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

Actual document definition is achieved using the UN/CEFACT Business Collaboration Models, or by some methodology external to ebXML but resulting in Schema definition (XSD or DTD) that an ebXML *Business Process Specification* can point to.

4. Choreography

The Business Collaboration Choreography describes the ordering and transitions between business transactions or sub collaborations within a binary collaboration. For example, in a UML tool this could be represented with a UML activity diagram. Actually, the choreography is specified in the ebXML *Business Process Specification Schema* using activity diagram concepts such as: start state, completion state, activities, forks, joins, decisions, transitions between activities, and guards on the transitions. However, it is beyond the scope of this document to specify a notation of a business collaboration.

5. Patterns

588 The ebXML *Business Process Specification Schema* provides a set of
589 unambiguous semantics, as asubset of UMM semantics, which enable us
590 to specify transactions and collaborations. Within these semantics the
591 user community has flexibility to specify an infinite number of specific
592 transactions and collaborations. The use of predefined patterns combines
593 this flexibility with a consistency that facilitates faster design, faster
594 implementation, and enables generic processing.

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

598 While the UMM transaction interaction patterns themselves are not part of
599 the ebXML BPSS technical specification, all the security and timing
600 parameters required to express the pattern properties are provided as
601 attributes of elements in the *Business Process Specification Schema*.

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

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

608

609 *5.10 How to use the ebXML Business Process Specification* 610 *Schema*

611 The ebXML *Business Process Specification Schema* should be used
612 wherever ebXML compliant software is being specified to execute Business
613 Collaborations.

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

617 This section discusses

618 • How the ebXML *Business Process Specification Schema* fits in with
619 other ebXML specifications.

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

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

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

628 5.11 How ebXML *Business Process Specification Schema* is 629 used with other ebXML specifications

630

631 The ebXML *Business Process Specification Schema* provides the structure
632 and semantics, as a subset of UMM semantics of Business Collaboration
633 definitions.

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

636 As shown in Figure 5, a BPSS instance will reference, but not define, a set of
637 required Business Documents. Within a BPSS instance, Business Documents
638 are either defined by some external document specification, or assembled
639 directly or indirectly from lower level information structures called core
640 components. The assembly is based on a set of contexts, many of which are
641 provided by the business processes, i.e. collaborations that use the
642 documents in their document flows.

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

646

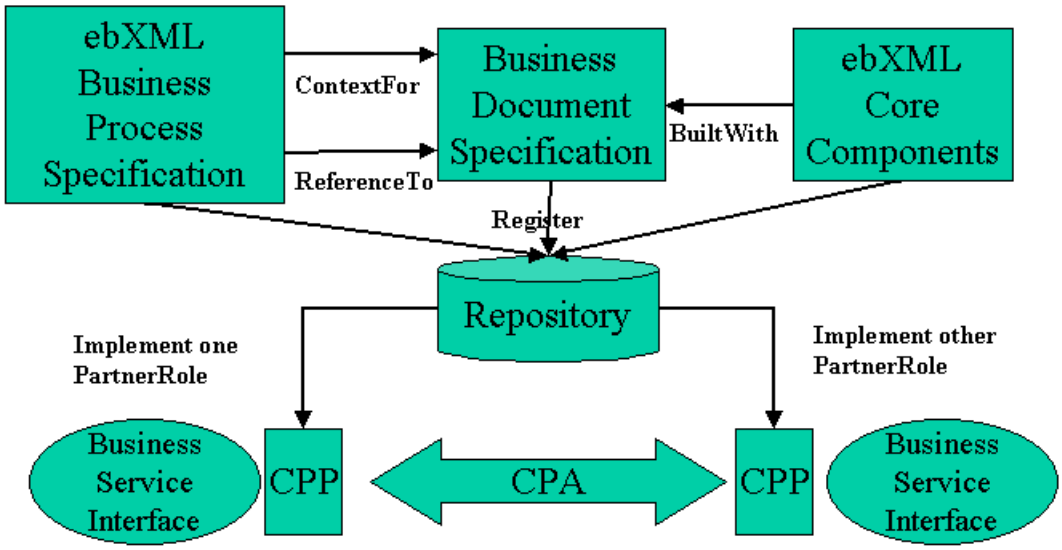


Figure 5: ebXML *Business Process Specification Schema* and other ebXML Specifications

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

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

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

When implementers want to establish trading partner Collaboration Protocol Profile and Agreement the *BPSS instance* document, or the relevant parts of it, are simply referenced by the CPP and CPA XML documents. ebXML CPP and CPA XML documents can reference business process specifications in XML such as an ebXML BPSS instance .

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.

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

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

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

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

The constructs listed above support the specification of fairly complex multi party collaborations. However, a BPSS instance may be as simple as a single Binary Collaboration referencing a single Business Transaction. This involves only numbers 1 through 3 above.

Note the ebXML BPSS technical specification does not specify any Business Process modeling methodology nor does it require the use of such methodology. Should a modeling methodology be needed, it is recommended to use the one of the UMM specification.

We have chosen a “drop ship” example which involves a buyer, a retailer, a vendor, and a credit organization. The order is placed by the buyer and fulfilled by the vendor. The credit authority makes sure that payments are made to appropriate creditors. We are using UML activity diagrams and use case diagrams to give a picturesc representation of this multi-party collaboration. This notation is non-normative and only here to help understand the example. It is used in a way that do not adhere to the UML semantics.

Figure 6 represents the overall multi-party collaboration. The conventation that we are using is such that an “activity” represents a binary collaboration between two roles. Since we have four roles represented here, we have adopted the following convention: the activity is placed in the swimlane of the role that starts the binary collaboration. The responding role is the one directly facing the activity. This is why the swimlane have different sizes.

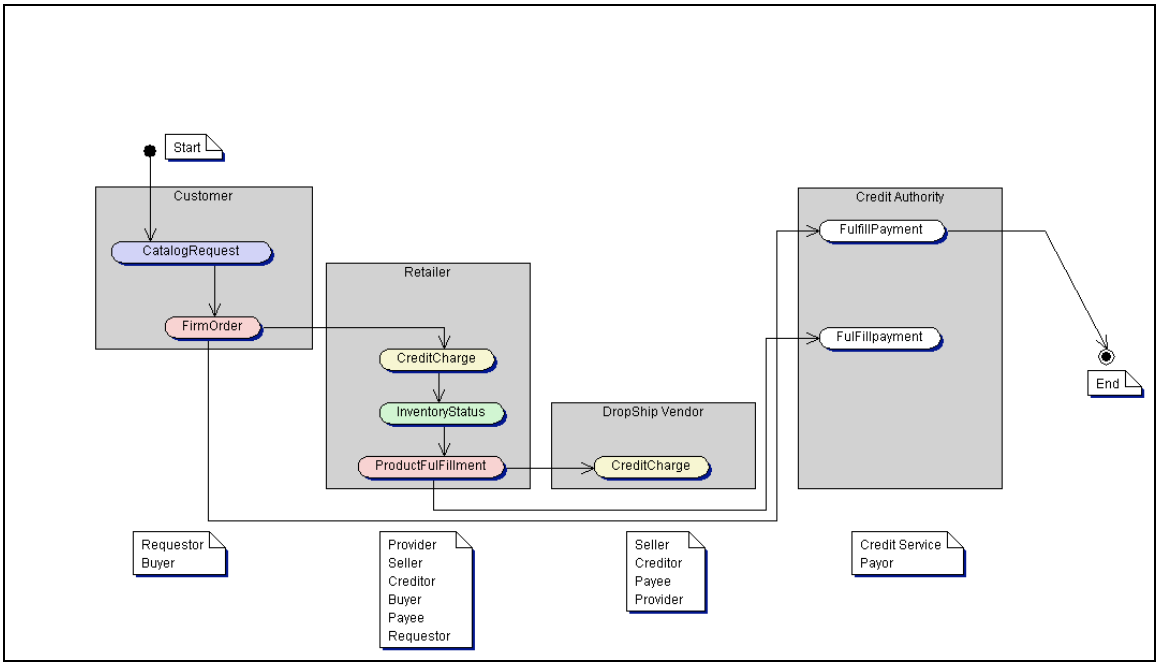


Figure 6. Representation of the "DropShip" multi-party collaboration with a UML activity diagram.

All binary collaboration in the example feature only one business transaction activity except two of them: Credit Charge and Product Fulfillment. These binary collaborations are represented on figure 7. with the same convention.

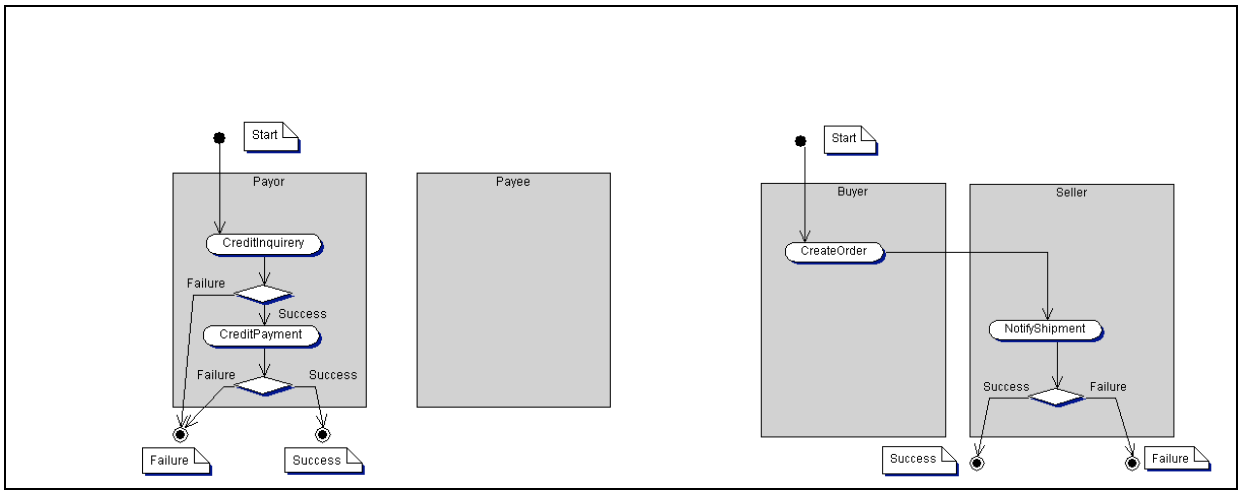


Figure 7. Representation of the "CreditCharge" and "ProductFulfillment" binary collaboriatons

Figure 8. Features all the binary collaboration definitions of the example (between abtract roles and business partner roles).

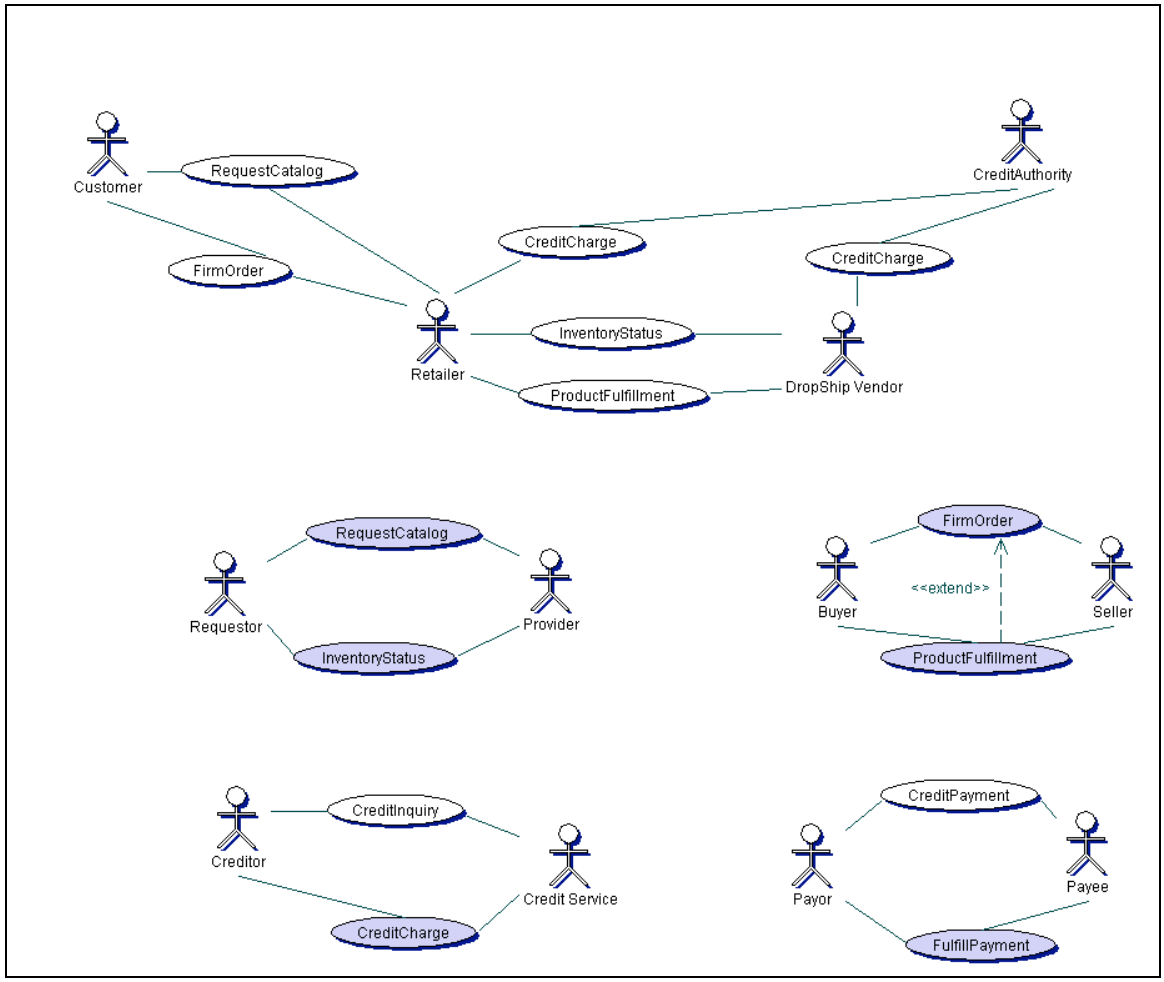


Figure 8. Multi-party and binary collaboration definitions of the example.

The complete XML is provided in Appendix A.

5.12.1 Packages and Includes

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

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

org.ebxml.transaction.order.ProcessPurchaseOrder

Which means that the *ProcessPurchaseOrder* business transaction is defined within the package *order*, which is itself, defined within the *transaction* package.

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

5.12.2 Substitution Sets

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

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

5.12.3 Specify a Business Transaction and its Business Document Flow

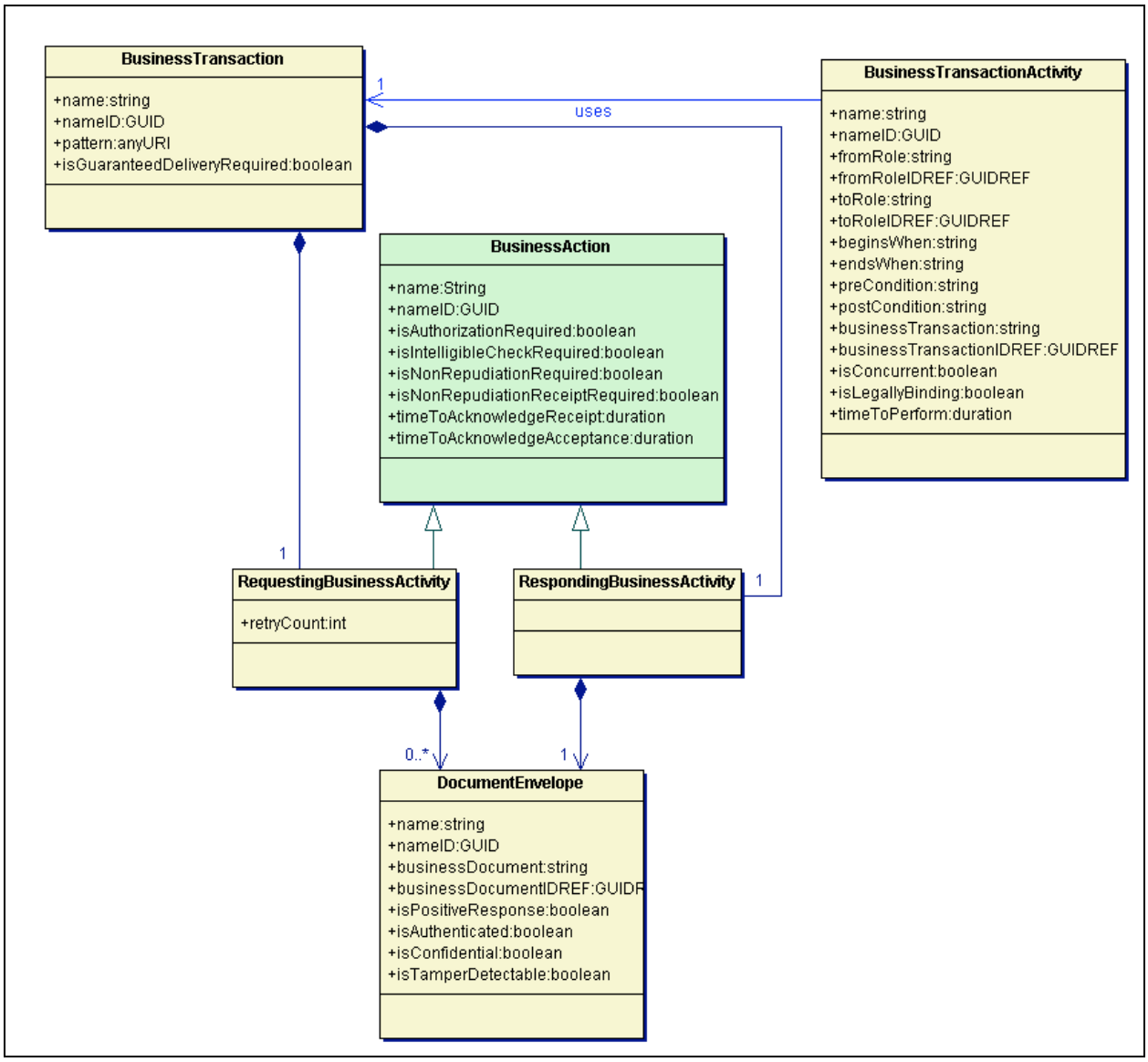


Figure 9 shows a part of the BPSS metamodel that defines the concept of Business Transaction.

Figure 9. UML Diagram of a Business Transaction

5.12.3.1 Key Semantics of a Business Transaction

A Business Transaction is an 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. A *Business Transaction* may support one or more Business Signals that govern the use and meaning of acknowledgements.

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

There is always a Request document flow.

A Business Transaction definition specifies whether a response document is required. This type of business transactions is typically associated with the formation of contracts or agreements. A Business Transaction with a request only is typically used for notifications.

An abstract superclass, *Business Action*, is the holder of attributes that are common to both Requesting Business Activity and Responding Business Activity. This element is abstract, it is does not appear in ebXML BPSS instances.

5.12.3.2 Sample syntax

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

```
<BusinessTransaction name="Catalog Request">
      <RequestingBusinessActivity name="requestCatalog"
               <DocumentEnvelope
            businessDocument="Catalog Request"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name="sendCatalog">
         <DocumentEnvelope
            isPositiveResponse="true"
            businessDocument="Catalog" />
      </RespondingBusinessActivity>
</BusinessTransaction>
```

Business signals acknowledging the document flow may be associated with each document flow .These acknowledgment signals are not specified explicitly however, two Business Transaction parameters specify whether the signals are required or not.

Figure 10 presents the possible Document Flows and business signals within a Business Transaction.

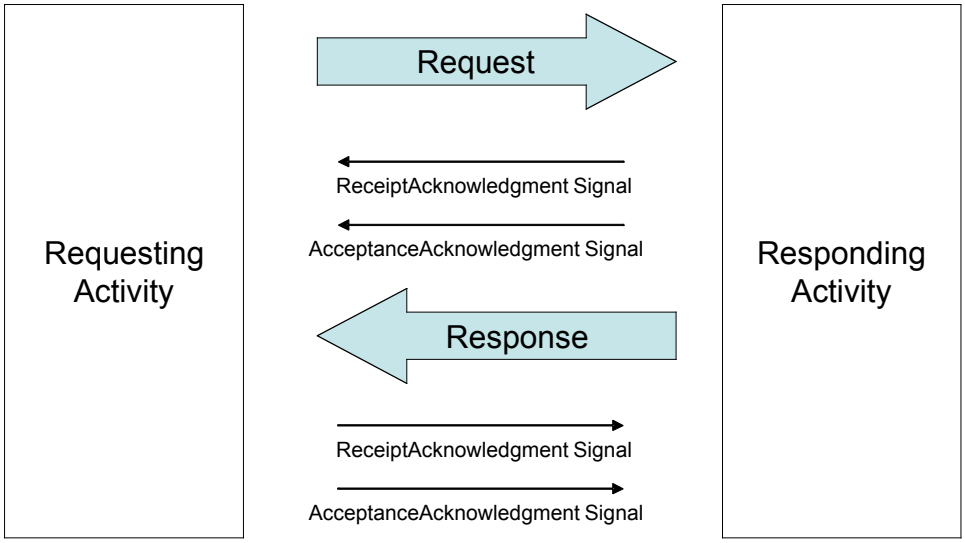


Figure 10. Possible document flows and signals and their sequence

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

The pattern of a *Business Transaction* may be used to specify whether a Receipt Acknowledgement and/or an Acceptance Acknowledgement signal are required. If the *pattern* attribute is not used, a non null value in the timeToAcknowledgeReceipt and timeToAcknowledgeAcceptance will mean that these signals must be issued by the corresponding party. Business transaction protocol signals are independent from lower protocol and transport signals such as reliable messaging.

The Receipt acknowledgement business signal, if used, signals that a message (Resquest or Response) has been properly received by the ebXML Business Service Interface software component. The property ***isIntelligibleCheckRequired*** allows partners to agree that a message should be confirmed by a Receipt acknowledgement only if it is also legible. Legible means that it has passed structure/ schema validity check. The content of the receipt and the legibility of a message (if required) are reviewed *prior* to the processing of the Business Document or the evaluation of condition expressions in the message's business documents or document envelope.

The Acceptance Acknowledgement business signal, if used, signals that the message received (Request or Response) has been accepted for business processing by the receiving application, or a receiving business application proxy. This is the case if the contents of the message's business documents and document envelope have passed a business rule validity check. These business rules are not necessarily specified as part of the collaboration. The state of each party is considered to be aligned when the receiving application (in general unknown to the other party) has signaled, *via* the BSI and an Acceptance Acknowledgement, that the business document has been successfully processed. Note that this acknowledgement is non-substantive, and simply indicate that the receiving party has reached a satisfactory state. If for any reason, the application could not process

the business document, the sending party should be notified via a negative Acceptance Acknowledgement signal such that is can transition to a meaningful “internal” business state. For instance, a Purchase Order could not be considered in the “sent” state, unless the other party had sent the corresponding Acceptance Acknowledgment. The substantive response would come after the signal indicating whether the order had been Accepted or Rejected.

Failure to send either signal, when *required* (by specifying a timeout value in *timeToAcknowledgeReceipt* or timeToAcknowledgeAcceptance), will result in the transaction being null and void, and therefore will prevent to reach any "success" end state (protocol or business) that would have depended on receipt of a business document satisfying the associated *timeToPerform*. In order for a business transaction activity instance to reach a “success” state at run-time, the following things would need to happen:

- no timeout would have occurred (signals or response)
- no signal can have a negative content
- the response document sent to the requestor must be marked as isPositiveResponse = ‘true’ in the ebXML BPSS instance that specifies the business collaboration

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

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

The *isGuaranteedMessageDeliveryRequired* refers to the underlying messaging service used to implement the business transaction protocol. The business transaction protocol is designed to achieve state alignment between both parties involved in the transaction and signals to the sending party the successful processing of the business documents, request or response, by the receiving application, whatever it might be. However, to achieve this result, the business transaction protocol shall be implemented on top of a reliable messaging service that provides guaranteed message delivery at the transport level. If the sending party was not guaranteed that its message or in particular signal reached the intended recipient, it could never be sure that the other party state is aligned with its own state. Since a signal structure is fixed there is no ambiguity about the BSI processing it and understanding its meaning provided you know that it reached its destination, unlike a request or response which could have

an invalid structure or content. In the case where the business transaction does not need to guarantee processing by the receiving application this condition can be relaxed and regular messaging services may be used.

Note that we can only guarantee the successful synchronization of state between two parties if reliable messaging is used and if the business transaction is defined to use the request and response acceptance acknowledgement signals, which guarantee that the corresponding business documents were processed by the respective applications.

5.12.3.3 Sample syntax

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

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

```
<BusinessTransaction
     name="CreateOrder"
     nameID="122A3DD33"
     isGuaranteedDeliveryRequired="true">
      <RequestingBusinessActivity
          name="sendOrder"
          nameID="122A3E833"
          isNonRepudiationReceiptRequired="false"
          isNonRepudiationRequired="false"
          timeToAcknowledgeAcceptance="P1H"
          timeToAcknowledgeReceipt="P1H">
            <DocumentEnvelope
               businessDocument="Purchase Order"
               businessDocumentIDREF="122A3F613"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity
          name="sendPOAcknowledgement"
          nameID="122A3E863"
          isNonRepudiationReceiptRequired="false"
          isNonRepudiationRequired="false"
          timeToAcknowledgeReceipt="P1D">
            <DocumentEnvelope
               isPositiveResponse="false"
               businessDocument="Reject Order"
               businessDocumentIDREF="122A3F8E3"/>
            <DocumentEnvelope
               isPositiveResponse="true"
               businessDocument="Accept Order"
               businessDocumentIDREF="122A3F6C3"/>
      </RespondingBusinessActivity>
</BusinessTransaction>
```

Note that duration are expressed using the standard duration type from the W3C’s XML Schema specification. For instance “P1D” means that we are specifying a “period” of 1 day.

5.12.3.4 Specifying Business Document flows

Request document flows and response document flows contain Business Documents that pertain to the *Business Transaction* request and response. The model for this is shown in Figure 11. Business Documents have varying structures. Business signals, however always have the same structure, defined once and for all as part of the ebXML *Business Process Specification Schema* technical specification.

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

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

Each Document Envelope carries exactly one primary Business Document.

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

992

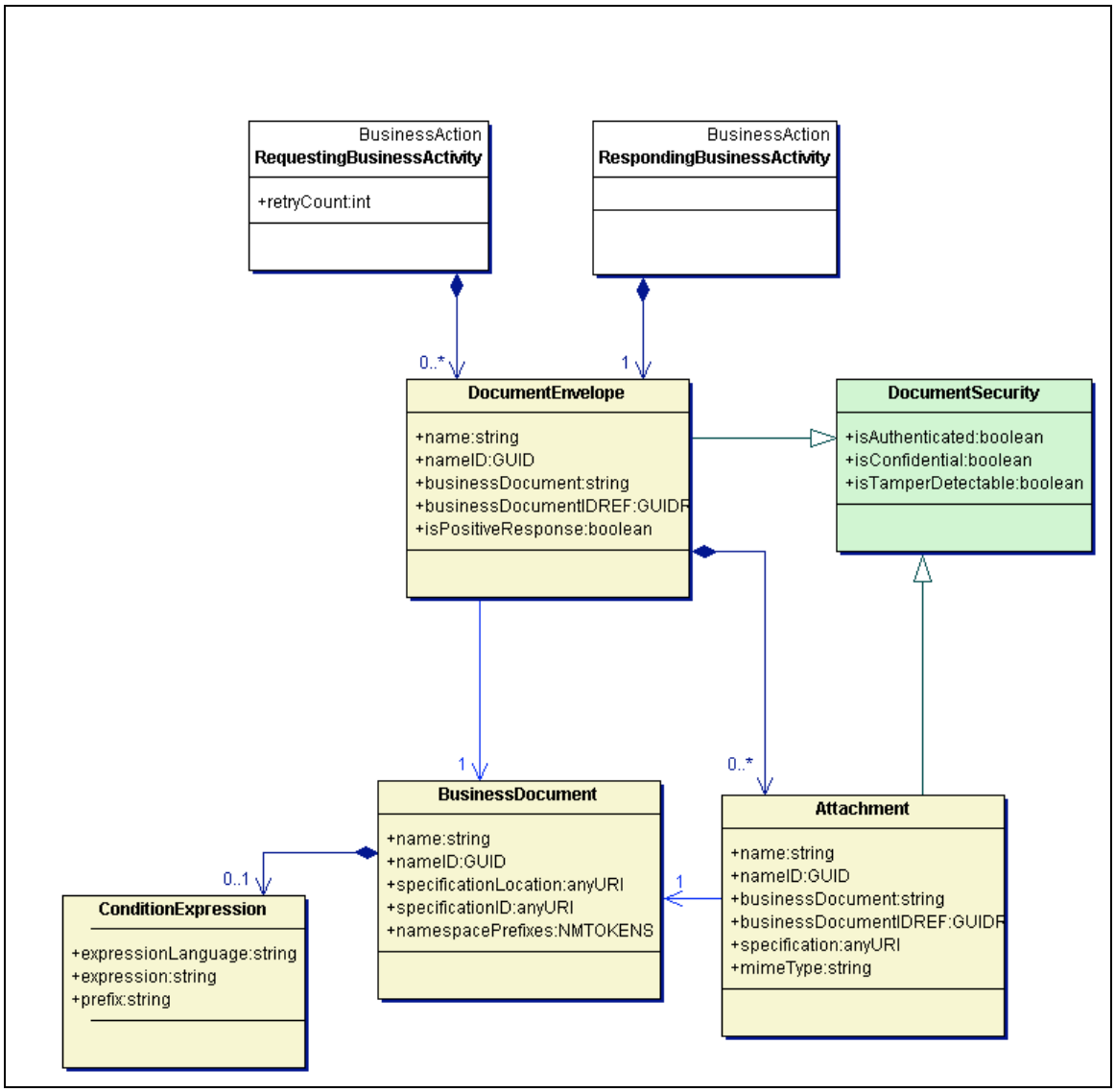


Figure 11. UML Diagram of document flow

5.12.3.5 Sample syntax

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

```
<BusinessDocument
      name="Credit Request"
      nameID="122A3F613C "
      specificationLocation="http://.../creditRequest.xsd
"
      specificationID="http://... /creditRequest.xsd"
      namespacePrefixes="fix">
</BusinessDocument>

<!-- The following two documents refer to the same
physical document, however, by their content as evaluated
at run-time, they are logically different -->
<BusinessDocument
      name="Credit Denied"
      nameID="122A3F8E3"
      specificationLocation="http://.../creditResponse.xs
d"
      specificationID="http://.../creditResponse.xsd"
      namespacePrefixes="fix">
      <ConditionExpression
            expressionLanguage="XPATH 1.0"
            expression="//@CreditResponse='denied'"
            prefix="fix"/>
</BusinessDocument>

<BusinessDocument
      name="Credit Approved"
      nameID="122A3F6C3"
      specificationLocation="http://.../creditResponse.xs
d"
      specificationID="http://.../creditResponse.xsd"
      namespacePrefixes="fix">
      <ConditionExpression
            expressionLanguage="XPATH 1.0"
            expression="//@CreditResponse='approved'"
            prefix="fix"/>
</BusinessDocument>

<BusinessDocument
      name="Credit Rating"
      nameID="122A3F8E4"
      specificationID="http://.../creditRating.id">
</BusinessDocument>

<BusinessTransaction
      name="Check Credit"
      nameID="122A3DD33"
      isGuaranteedDeliveryRequired="true">
      <RequestingBusinessActivity
            name="checkCredit"
            nameID="122A3E833"
```

```
                        isAuthorizationRequired="true"
                        isIntelligibleCheckRequired="true"
                        isNonRepudiationReceiptRequired="true"
                        isNonRepudiationRequired="true"
                        timeToAcknowledgeAcceptance=" PT30S"
                        timeToAcknowledgeReceipt=" PT10S">
                        <DocumentEnvelope
                              isAuthenticated="persistent"
                              isConfidential="persistent"
                              isTamperDetectable="persistent"
                              businessDocument=" Credit Request"
                              businessDocumentIDREF="122A3F613C"/>
                  </RequestingBusinessActivity>

                  <RespondingBusinessActivity
                        name="confirmCredit"
                        nameID="122A3E863"
                        isAuthorizationRequired="true"
                        isIntelligibleCheckRequired="true"
                        isNonRepudiationReceiptRequired="true"
                        isNonRepudiationRequired="true"
                        timeToAcknowledgeReceipt="PT10S">
                        <DocumentEnvelope
                              isPositiveResponse="false"
                              isAuthenticated="persistent"
                              isConfidential="persistent"
                              isTamperDetectable="persistent"
                              businessDocument="Credit Denied"
                              businessDocumentIDREF="122A3F8E3"/>
                        <DocumentEnvelope
                              isPositiveResponse="true"
                              isAuthenticated="persistent"
                              isConfidential="persistent"
                              isTamperDetectable="persistent"
                              businessDocument="Credit Approved"
                              businessDocumentIDREF="122A3F6C3">
                              <Attachment
                                    name=”Credit Report”
                                    mimeType=”XML”
                                    businessDocument=”Credit Rating”
                                    businessDocumentIDREF="122A3F8E4"
                                    isConfidential=”none”
                                    isTamperDetectable=”none”
                                    isAuthenticated=”none”>
                              </Attachment>
                        </DocumentEnvelope>
                  </RespondingBusinessActivity>
            </BusinessTransaction>
```

See section 7.5.5. for a discussion on document security parameters.

5.12.4 Specify a Binary Collaboration

Figure 12 shows part of the metamodel of a binary collaboration.

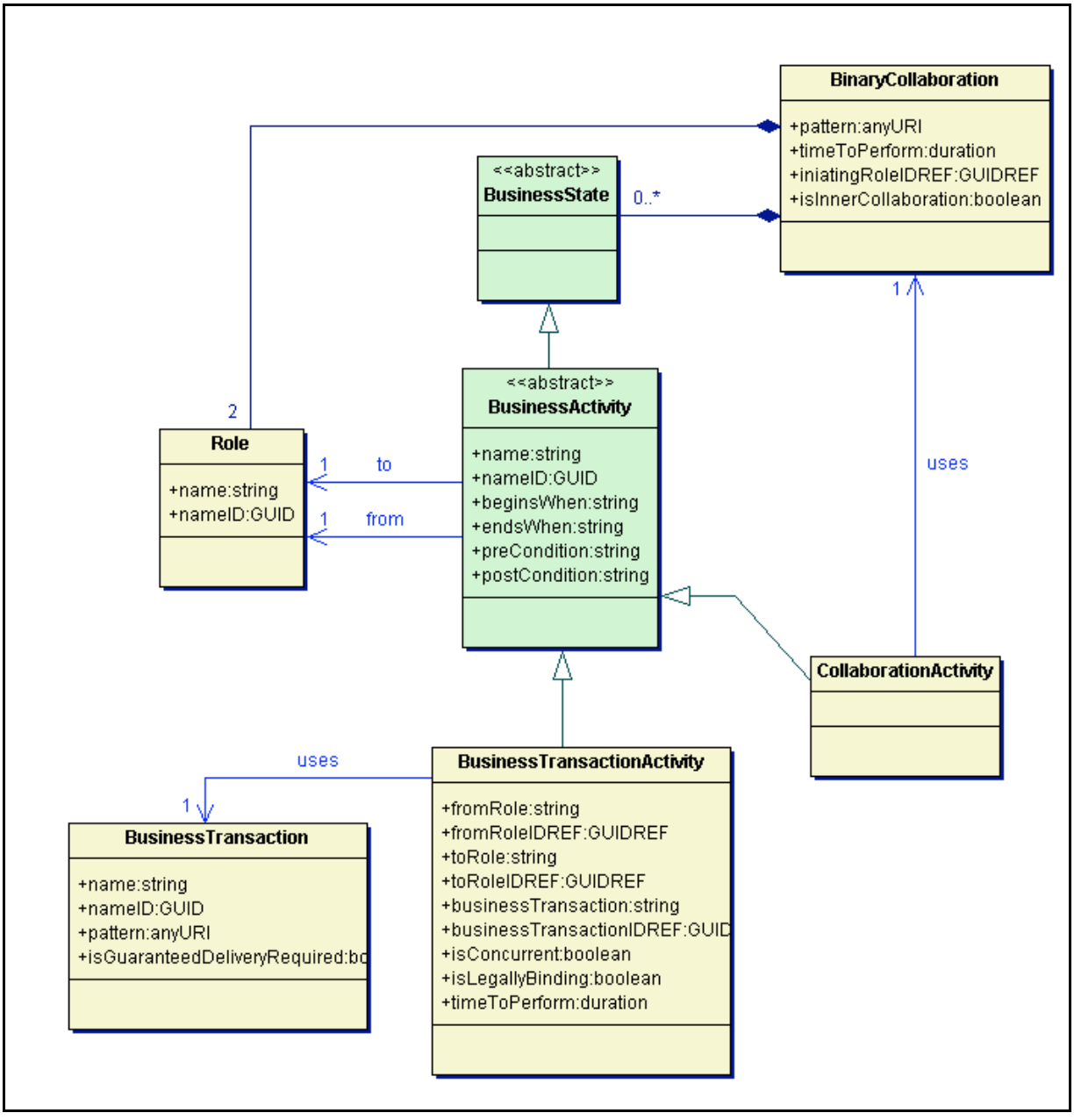


Figure 12. UML Diagram of a Binary Collaboration

5.12.4.1 Key Semantics of a Binary Collaboration

A *Binary Collaboration* is always defined between two roles. One of the roles is initiating the collaboration. This is the role, which sends the first message (i.e. Request) of the first *Business Transaction Activity*. This attribute is used to "bind" the roles of an inner *Collaboration Activity* to the parent *Binary Collaboration* roles. Even if this role is not know until run-time, we can still specify a "logical" initiating role. In that case, the initiating role of the parent binary collaboration definition will be bound to the initiating role of the inner binary collaboration definition.

It is critical that the *Role nameID* be unique with respect to a *Binary Collaboration* definition even if role names are identical for two *Binary Collaboration* definitions. This means that two binary collaboration may never have the same physical role, but share a "logical" role.

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

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 Transaction definitions can be associated to any number of Business Transaction Activity elements. This means that the same *Business Transaction* can be performed by multiple *Business Transaction Activities* in different *Binary Collaborations*, or by multiple *Business Transaction Activities* in the same *Binary Collaboration*, sometimes with opposite roles. For instance a "Cancel Purchase Order" *Business Transaction* could be used by two Business Transaction Activities, which can be performed by either party, meaning that after a purchase order has been accepted, either party could cancel it (for a certain period of time) using the same business document interchange.

A *Collaboration Activity* is the performance of a *Binary Collaboration*, within another *Binary Collaboration*. *Binary Collaboration definitions* are re-useable relative to Collaboration Activity. The same *Binary Collaboration* can be performed by multiple *Collaboration Activities* in different *Binary Collaborations*, or by multiple *Collaboration Activities* in the same *Binary Collaboration*. A binary collaboration definition may be restricted to be an "inner collaboration" only via the the boolean attribute *isInnerCollaboration*. In this case, the binary collaboration definition can only be initiated as part of a *Collaboration Activity* and cannot be initiated by itself.

Business Transaction Activity and *Collaboration Activity* may define business rules with the *beginsWhen*, *endsWhen*, *preCondition* and *postCondition* attributes. These attributes do not have a specific syntax as part of this specification, so the current type is string. Because these expressions cannot be generally executed by an ebXML infrastructure, in the current release of the ebXML BPSS

technical specification, they are considered to have a “documentation” purpose. In particular they cannot be used to specify any part of the choreography of the collaboration. In future releases they will play a role along with transitions and pseudo-states. The semantics of beginsWhen and endsWhen indicate that the corresponding business activity needs to be started or ended as soon as the expression in the attribute value is true. PreConditions and postConditions indicate that the corresponding business activity may start only if the corresponding expressions are true.

When performing a *Binary Collaboration* within a *Binary Collaboration* there is an implicit relationship between the roles at the two levels. Assume that *Binary Collaboration* X is performing *Binary Collaboration* Y through Collaboration Activity Q. Binary Collaboration X has the following roles: Customer and Vendor. In Collaboration Activity Q we assign Customer to be the initiator, and Vendor to be the responder. Binary Collaboration Y has the following roles: Buyer and Seller and a Business Transaction Activity where Buyer is the initiator and Seller the responder. We have now established a role relationship between the roles Customer and Buyer because they are both initiators in activities in the related performing and performed Binary Collaborations.

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

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

5.12.4.2 Sample syntax

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

```
<BinaryCollaboration
      name="Firm Order"
      nameID="122A38D93"
      initiatingRoleIDREF=”122A38DA3”
      timeToPerform="P1D">
      <Role
            name="buyer"
            nameID="122A38DA3"/>
      <Role
            name="seller"
            nameID="122A38DA5"/>
      <Start
            toBusinessState=”Place Order”
            toBusinessStateIDREF=”122A39C23” />
      <BusinessTransactionActivity
            name="Place Order"
            nameID="122A39C23"
            businessTransaction="Create Order"
            businessTransactionIDREF="122A3DD33"
            fromRole="buyer"
            fromRoleIDREF="122A38DA3"
            toRole="seller"
            toRoleIDREF="122A38DA5"
            isConcurrent="true"
            isLegallyBinding="false"
            timeToPerform="P2H"/>
      <Failure
            fromBusinessState="Place Order"
            fromBusinessStateIDREF="122A39C23"
            conditionGuard="AnyProtocolFailure"/>
      <Success
            fromBusinessState="Place Order"
            fromBusinessStateIDREF="122A39C23"
            conditionGuard="BusinessSuccess |
            BusinessFailure"/>
</BinaryCollaboration>
```

5.12.5 Specify a MultiParty Collaboration

Figure 13 shows the metamodel a multiparty collaboration

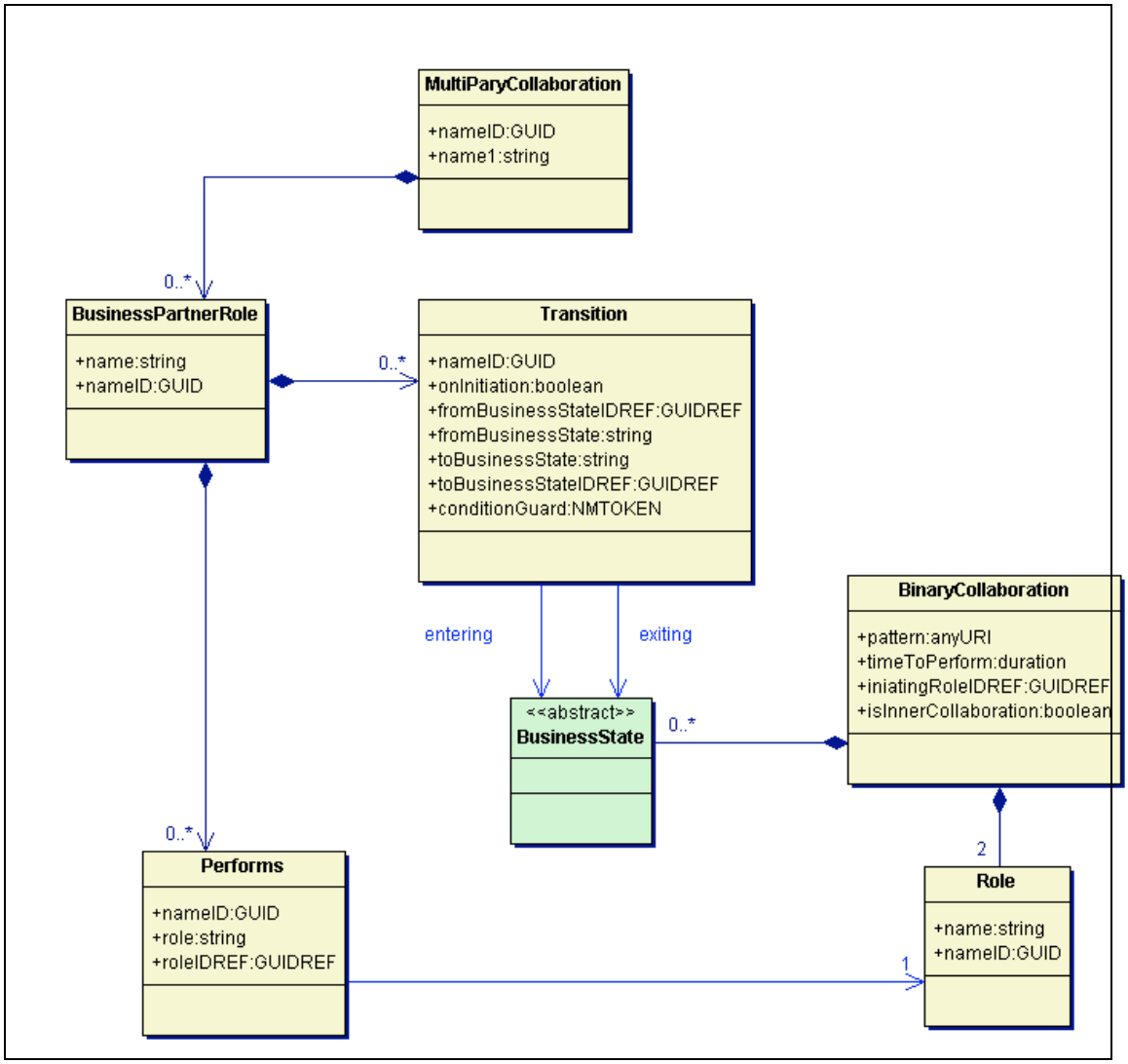


Figure 13: UML Diagram of a MultiParty Collaboration

5.12.5.1 Key Semantics of a Multiparty Collaboration

A Multiparty Collaboration is specified as a synthesis of Binary Collaborations definitions.

A Multiparty Collaboration is defined as a list of "*Business Partner Roles*".

Each *Business Partner Role Performs* one or more *Role* several binary collaboration definitions. Note that the Binary Collaboration to Role relationship is navigable, which means that a Perfoms element uniquely identify a Binary Collaboration and a Role within this Binary Collaboration definition. It is often the case that a business partner (role) plays several "roles" in a multi-party collaboration. For instance, a distributer role, would play both the roles of buyer and seller in a purchasing collaboration involving a customer (buyer), distributor (seller, buyer) and a manufacturer (seller).

This association between a *Business Partner Rol*e and a specific *Role* in a specific *Binary Collaboration* is specified by the *Performs* element and is what constitute the synthesis of *Binary Collaborations* into *Multiparty Collaborations*.

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

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

5.12.5.2 Sample syntax

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

```
<MultiPartyCollaboration name="DropShip">
      <BusinessPartnerRole name="Customer">
            <Performs role="requestor"
roleIDREF="1122B1"/>
            <Performs role="buyer" roleIDREF="1122B2"/>
            <Transition
                  fromBusinessState="Catalog Request"
                  toBusinessState="Create Order"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="Retailer">
            <Performs role="provider"
roleIDREF="2211A1"/>
            <Performs role="seller" roleIDREF="1122B3"/>
            <Performs role="creditor"
roleIDREF="9122B1"/>
            <Performs role="buyer" roleIDREF="1122B2"/>
            <Performs role="payee" roleIDREF="6122B1"/>
```

```
                  <Performs role="requestor"
            roleIDREF="1122B1"/>
                  <Transition
                        fromBusinessState="Create Order"
                        toBusinessState="Check Credit"/>
                  <Transition
                        fromBusinessState="Check Credit"
                        toBusinessState="Credit Payment"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="DropShip Vendor">
                  <Performs role="seller" roleIDREF="1122B3"/>
                  <Performs role="payee" roleIDREF="6122B1"/>
                  <Performs role="creditor"
            roleIDREF="9122B1"/>
                  <Performs role="provider"
            roleIDREF="2211A1"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="Credit Authority">
                  <Performs role="credit service"
                              roleIDef="8122B1"/>
                  <Performs role="payor" roleIDREF="7122B1"/>
            </BusinessPartnerRole>
      </MultiPartyCollaboration>
```

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

5.12.6 Specify a Choreography

Figure 14 shows the metamodel of a choreography.

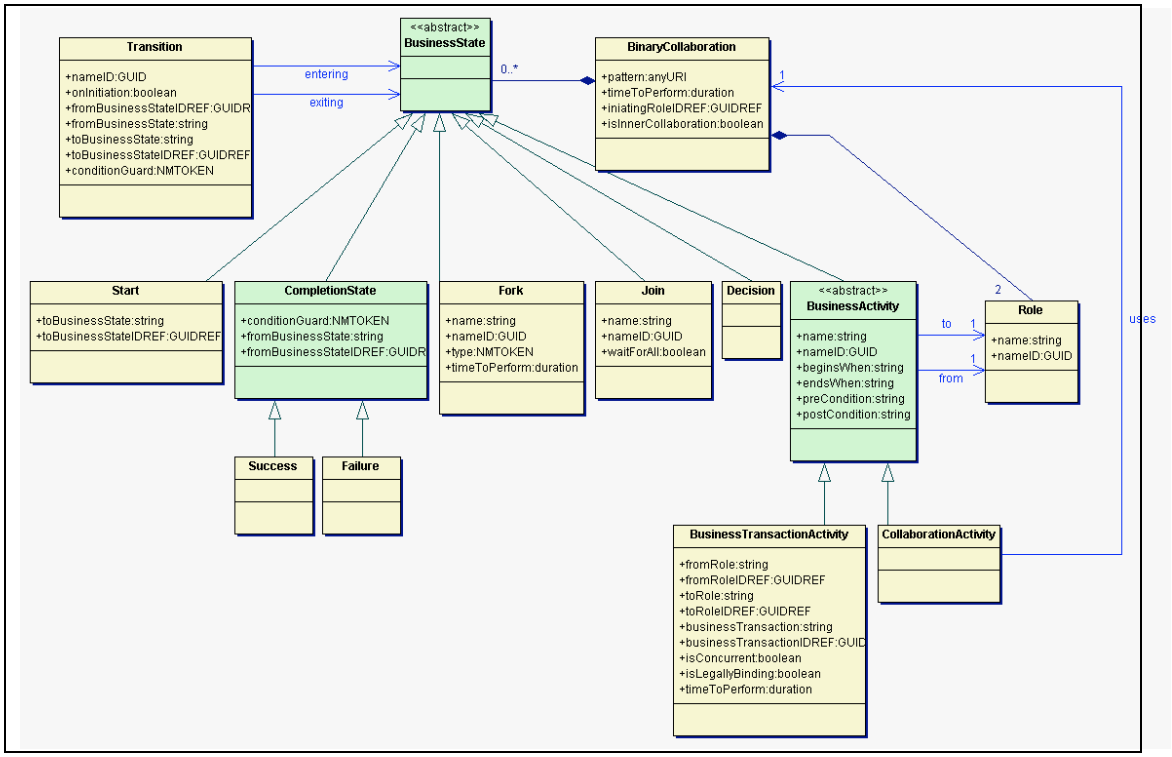


Figure 14: UML Diagram of a Choreography

5.12.6.1 Key Semantics of a Choreography

A Choreography is an ordering of Business Activities within a Binary Collaboration. The purpose of a Choreography is to specify which Business Transaction Activity and/or Collaboration Activity should happen at any point in time. As a result, the specification of choreography definition and the business transaction protocol defines unambiguously which message (DocumentEnvelope or Signal) is expected by any of the parties at any point in time.

The choreography is specified in terms of Business States, and transitions between those Business States. When such a Business State is a Business Activity, a transition happens between the Business Activity completion state and the start state of the following Business Activity. When a transition is validated, it does not mean that the target Business Activity would start immediately. Instead, it means that the Business Activity is “enabled” and the initiating party may now send the request whenever appropriate, provided that it remains within the *timeToPerform* of the binary collaboration.

A Business Activity is an abstract kind of Business State. Its two subtypes Business Transaction Activity and Collaboration Activity are

concrete Business States. The business collaboration can be said to be in the state of performing a given business activity. Once a business activity complete a transition from this business activity is navigated to another business activity or pseudo-state. A message shall either initiate a collaboration or advance its state.

There are a number of auxiliary kinds of Business States that facilitate the choreographing of Business Activities. These include a Start state, a Completion state (which comes in a Success and Failure flavor), a Fork state, a Join state and a Decision state. These are all equivalent to diagramming artifacts on a UML activity diagram, however, the semantics are not exactly the same. An XOR value in the type attribute of a fork means that only one Business State of the fork will be allowed to be reached. All the other will become invalid as soon as one of the business state is reached (e.g. a Business Transaction Activity starts). An OR value will mean that any business activity pointed to by a transition coming from the fork might be initiated. These business activities may occur in parallel. Note that it is not important to specify the order in which condition expression on a transition coming from a fork will be evaluated. It is merely the order in which the request of the business transaction activities will arrive that will determine the order in which the condition expression need to be evaluated. A fork has a timeToPerform attribute. At the end of this time interval, the state of the Binary Collaboration will automatically be moved to its corresponding join. This feature is useful in cases where the business activities are optional. For instance a Cancel Purchase Order and Change Purchase Order business transaction activity could be defined as part of a Fork/Join control block. However, most often none of these activity would happen. If any given Business Transaction Activity within the Fork/Join pair is has not reached its completion state, the BSI will generate a corresponding timeout exception. As a well formed rule, the timeToPerform of a fork can not be less that any timeToPerform of its business activities. The waitForAll attribute of the join will indicate that all transitions coming into the join shall be executed in order for the collaboration to reach the join pseudo-state (AND-join), by default, the join is an AND-join. When this parameter is set to false, it is an OR-join. The BSI will generate a timeout exception if an OR-join is reached while a Business Activity has not reached its completion state. The semantics of fork and join are such that for instance a fork may be defined without a corresponding join. In this case, the timeToPerform attribute shall not be used. It must only be used in the case where all outgoing transitions from the fork have incoming transitions to the join.

Fork	Join	Comments
OR	WaitforAll (true)	This models the behavior of an AND-fork and AND-Join
OR	WaitforAll (false)	If timeout is null, should rather use XOR as the join will happen on the first transition reaching the join state
XOR	WaitforAll (true)	This combination is forbidden (would lead to a dead lock)

XOR	WaitforAll (false)	Only one path between the fork and join will be allowed to happen
timeToPerform >0	Any value	The join happens when timeToPerform is reached.

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

The Success and Failure elements represent an aggregation of a state and a transition to this particular state. This transition like regular transitions can be guarded by a conditionGuard. The conditionGuard can be used to indicate that a binary collaboration ends in success or failure based on the fact that the last business transaction activity response is a business document of a particular type, or based on the content of the response. It is important to note that the success or failure of the collaboration does not affect the success or failure of the individual business transaction activities, which compose the binary collaboration. In particular, the nature of the commitments is not changed when the collaboration ends in a specific state. The success or failure of a collaboration is rather an indication, which can be reported on, or acted upon to initiate other collaborations. If several completion states are specified within a collaboration definition, the business collaboration run-time instance state is “complete” as soon as one of the completion state is reached. It is the responsibility of the designer to ensure that all completion states are mutually exclusive and that once one of them is reached there are no further Business Activity open. A timeout exception will be generated by the BSI in such a case.

A Transition can also be used to create nested BusinessTransaction-Activities. A nested BusinessTransactionActivity is enabled when a transition flows from a parent BTA to the nested BTA and this transition is marked onInitiation = ‘true’. In this case, the transition is enabled after the receipt of the request in the parent transaction but after the request has been acknowledged appropriately if applicable. At this point, the second activity is performed before returning sending the response to the original requestor. No “return” transition is

specified. If more than one transaction ought to be executed they must be specified within a collaboration activity. There are no possible outgoing transitions fom a nested activity unless it is associated to an exception. If the activity terminates normally, the thread of control is handed back to the parent activity. The flag 'onInitiation' in Transition is used for this purpose. Nested *Business Transaction Activity* are often found within a multiparty collaboration. In essence a Role in one Binary Collaboration receives a request, then turns around and becomes the requestor in other Binary Collaboration before coming back and sending the response in the first Binary Collaboration.

isConcurrent is a parameter that governs the flow of transactions. Unlike the security and timing parameters it does not govern the internal flow of a transaction, rather it determines whether at run-time multiple instances of that business transaction activity can be 'open' at the same time within any collaboration instance perfomed between any two partners. As a result, when *isConcurrent* is set to false, the BSIs of each party are responsible for serializing these business transaction activities.

5.12.6.2 Sample syntax

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

```
<BinaryCollaboration
      name="Product Fullfillment"
      nameID="122A38D93"
      Role=”233A38DA3”
      timeToPerform="P3D">
      <Role
            name="buyer"
            nameID="122A38DA3"/>
      <Role
            name="seller"
            nameID="122A38DA5"/>
      <Start
            toBusinessState=”Create Order”
            toBusinessStateIDREF=”122A39C23”/>
      <BusinessTransactionActivity
            name="Create Order"
            nameID="122A39C23"
            businessTransaction="Create Order"
            businessTransactionIDREF="122A39C24"
            fromRole="buyer"
            fromRoleIDREF="122A38DA3"
            toRole="dealer"
            toRoleIDREF="122A38DA5"
            isConcurrent="true" isLegallyBinding="false"
            timeToPerform="P1H"/>
      <BusinessTransactionActivity
            name="Notify Shipment"
            nameID="122A39CA3"
```

```
                businessTransaction="Notify Shipment"
                businessTransactionIDREF="122A39CA4"
                fromRole="seller"
                fromRoleIDREF="122A38DA3"
                toRole="buyer"
                toRoleIDREF="122A38DA3"
                isConcurrent="true" isLegallyBinding="false"
                timeToPerform="P2D"/>
        <Success
                fromBusinessState="Test Success"
                fromBusinessStateIDREF="54654B789"
                conditionGuard="Success"/>
        <Failure
                fromBusinessState="Test Success"
                fromBusinessStateIDREF="54654B789"
                conditionGuard="AnyProtocolFailure |
                BusinessFailure"/>
        <Failure
                fromBusinessState="Test Order Accepted"
                fromBusinessStateIDREF="54654B567"
                conditionGuard="AnyProtocolFailure |
                BusinessFailure">
                        <ConditionExpression
                                expressionLanguage=
                                "DocumentEnvelopeNotation"
                                conditionExpression=
                                "Reject Order"/>
        </Failure>
        <Transition
                fromBusinessState="Test Order Accepted"
                fromBusinessStateIDREF="54654B567"
                toBusinessState="Notify Shipment"
                toBusinessStateIDREF="122A39CA3"
                conditionGuard="Success">
                        <ConditionExpression
                                expressionLanguage=
                                "DocumentEnvelopeNotation"
                                conditionExpression=
                                "Accept Order"/>
        </Transition>
        <Transition
                fromBusinessState="Create Order"
                fromBusinessStateIDREF="122A39CA3"
                toBusinessState="Test Order Accepted"
                toBusinessStateIDREF="54654B789"/>

        <Transition
                fromBusinessState=" Notify Shipment "
                fromBusinessStateIDREF="122A39CA3"
                toBusinessState="Test Success"
                toBusinessStateIDREF="54654B789"/>
        <Decision
                name=”Test Success”
                nameID=”54654B789”/>
        <Decision
                name=”Test Order Accepted”
                nameID=”54654B567”/>

</BinaryCollaboration>
```

Note that all the completion states of this binary collaboration definition are mutually exclusives.

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

```
<Transition
      onInitiation="false"
      fromBusinessState="Update Repair Order"
      fromBusinessStateIDREF="122A39CA3"
      toBusinessState="Process Repair Order"
      toBusinessStateIDREF="122A39C23"
      conditionGuard="Success">
      <ConditionExpression
            expressionLanguage=”XPath 1.0”
            conditionExpression=
      ”//POAck[@status=’Reject’]”/>
</Transition>
```

Similarly, transitions can be defined between Business Activities of a multi-party collaboration.

```
<MultiPartyCollaboration name="DropShip">
      <BusinessPartnerRole name="Customer">
            <Performs role="requestor" roleIDREF="1122B1"/>
            <Performs role="buyer" roleIDREF="1122B2"/>
            <Transition
                  fromBusinessState="Catalog Request"
                  toBusinessState="Create Order"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="Retailer">
            <Performs role="provider" roleIDREF="2211A1"/>
            <Performs role="seller" roleIDREF="1122B3"/>
            <Performs role="creditor" roleIDREF="9122B1"/>
            <Performs role="buyer" roleIDREF="1122B2"/>
            <Performs role="payee" roleIDREF="6122B1"/>
            <Performs role="requestor" roleIDREF="1122B1"/>
            <Transition
                  fromBusinessState="Create Order"
                  toBusinessState="Check Credit"/>
            <Transition
            fromBusinessState="Check Credit"
            toBusinessState="Credit Payment"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="DropShip Vendor">
            <Performs role="seller" roleIDREF="1122B3"/>
            <Performs role="payee" roleIDREF="6122B1"/>
            <Performs role="creditor" roleIDREF="9122B1"/>
            <Performs role="provider" roleIDREF="2211A1"/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="Credit Authority">
            <Performs role="credit service"
      roleIDREF="8122B1"/>
            <Performs role="payor" roleIDREF="7122B1"/>
      </BusinessPartnerRole>
</MultiPartyCollaboration>
```

5.12.7 The whole model

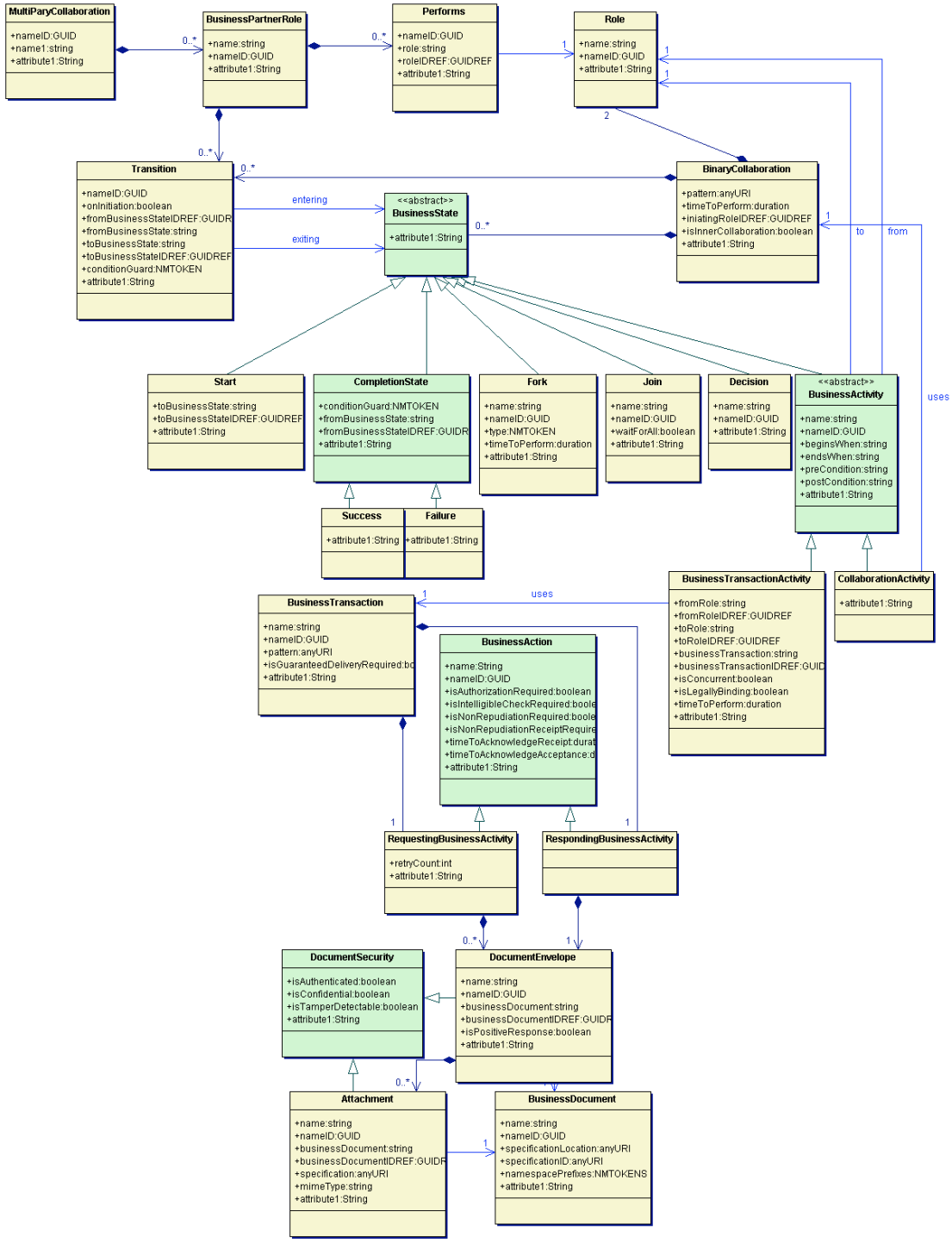


Figure 15. Overall UML Model of the ebXML Business Process Specification Schema

Figure 15 represents the complete metamodel of ebXML BPSS as a UML class diagram..

5.13 Core Business Transaction Semantics

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

The ebXML Business Transaction semantics, i.e. the rules and configuration parameters required for Business Service Interface software to predictably and deterministically execute ebXML Business Transactions, allows you to specify electronic commerce transactions that provide

- Interaction Predictability, i.e. have clear roles, clear transaction scope, clear time bounds, clear business information semantics, clear determination of success or failure. Each party can compute without ambiguity and the status of a transaction independently.
- Ability to create Legally Binding Contracts, i.e. the ability to specify that Business Transactions may be agreed to bind the parties. This concept is being deprecated as of this version.
- Nonrepudiation, i.e. may specify the keeping of artifacts to aid in legal enforceability.
- Authorization Security, i.e. may be specified to require authorization of parties performing roles.
- Document Security, i.e. may be specified to be authorized, authenticated, confidential, tamper detectable.
- Reliability, i.e. the ability to specify reliable delivery of Business Documents and signals.

Each of the above characteristics of the concept that we call an ebXML Business Transaction semantics is discussed in detail below.

These desirable characteristics are only applicable to ebXML *Business Transactions*, where an ebXML *Business Transaction* is a single request or single request / response pair only. A future version of this specification may extend the applicability of these characteristics to other types of electronic commerce transactions. In particular, we do not claim that the ebXML *Business Transaction* concept covers all possible electronic commerce transactions. For instance, a use case could involve an electronic commerce transaction that exchanges a request and two responses as a unit of work. If we would want to have similar properties, this kind of use cases would not be directly covered by this specification. The only way to handle such a use case would be to specify the electronic commerce transaction as a binary collaboration involving as many ebXML Business Transaction as necessary. The binary collaboration definition would then be specified in such a way to handle the individual ebXML Business Transaction exceptions and aggregate them into the electronic commerce transaction.

5.13.1 Interaction Predictability

All Business Transactions follow a very precisely prescribed flow, or a precisely defined subset there-of. The following is an overall illustration of this flow. It can be thought of as the state machine across the two business partners.

The goal of the Business Transaction Protocol is to synchronize the business state between two parties. As few resources can be shared between company boundaries, we must use such protocol to achieve the business state synchronization as recorded by each party enterprise systems.

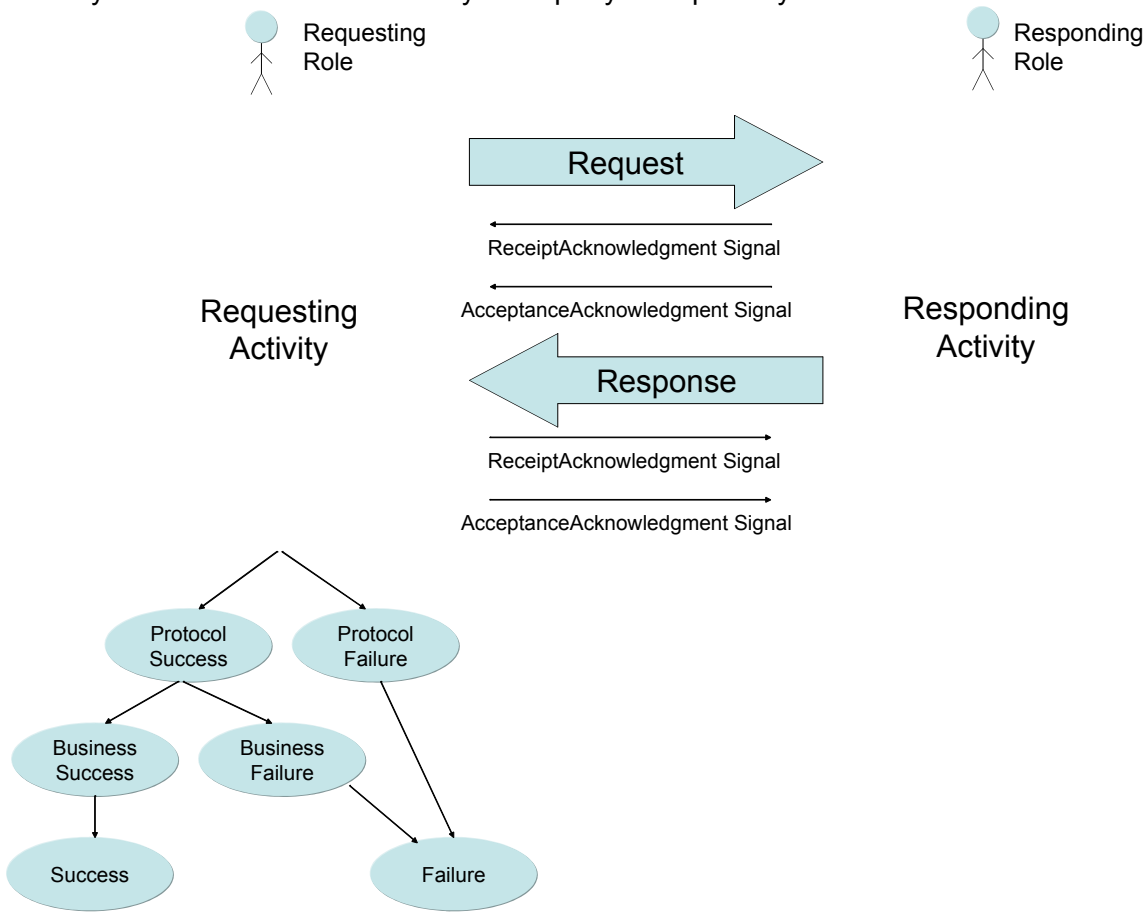


Figure 16: Schematic of core Business Transaction semantics.

Figure 16 does not assume any hierarchy in the way exceptions are generated or evaluated. It simply state that in oder to achieve a success state a business transaction activity complete with both a procotol and a business success. These exception are constantly evaluated by the BSI, and thrown as soon as detected.

If either a protocol or business failure occurs, the business transaction activity will be put into a failure state.

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

1. The Business Transaction is an atomic unit of work. All of the interactions in a business transaction must succeed or each party must not change their state.
2. A Business Transaction is conducted between two business partners playing opposite roles in the transaction. These roles are always the Requesting Role and the Responding Role.

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

5.13.1.1 Transaction Interaction Patterns

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

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

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

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

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

The specification of a business transaction may require each one of these signals independently of whether the other is required. Therefore there is a finite set of combinations. The UMM supplies the currently defined set of transaction patterns.

5.13.2 Creating legally binding contracts

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

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

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

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

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

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

5.13.3 Non-Repudiation

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

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

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

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

The other requires the receiver of a business document to send a signed receipt, which the original sender saves. This is the

isNonRepudiationOfReceiptRequired parameter in the requesting and responding business activity.

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

5.13.4 Authorization security

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

In order to invoke those rules, a party may specify *isAuthorizationRequired* on a requesting and/or a 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 [Role] to a list of allowed values previously supplied by that party.

isAuthorizationRequired is specified on the requesting and responding activity accordingly.

This concept is deprecated is as of this version. Its specification might change in a future release and is not required for an ebXML BPSS 1.1 compliant BSI infrastructure. In this version, a BSI would have no way to specify that an attempt has been made by an application or system to initiate a Business Transaction (therefore sending a request) and this application or system was not authorized to do so.

5.13.5 Document security

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

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

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

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

Persistent confidentiality is intended to preserve the confidentiality of the message such that only the intended party (application) can see it. The message shall remain in encrypted form after it is delivered to the MSH node

and will be decrypted only by the authorized application. S/MIME can be used to provide that functionality, independent of the transient confidentiality.

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

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

5.13.6 Reliability

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

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

5.13.7 Parameters required for CPP/CPA

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

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

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

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

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

5.14 Run time Business Transaction semantics

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

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

1. Detection of the opening of a transaction
2. Detection of transfer of control
3. Detection of successful completion of a transaction

a. Application of business rules expressed as schema definitions and *isPositiveResponse* for determination of success

4. Detection of failed completion of a transaction
   a. Detection of time-outs
   b. Detection of protocol exceptions
   c. Validation of the received response and identify if it was specified with isPositiveResponse = false

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

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

5.14.1 Timeouts

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

Here are the timeout parameters relative to the three response types:

Response required	Parameter Name and meaning of the timeout
Receipt Acknowledgement	*timeToAcknowledgeReceipt*
	The time a responding or requesting role has to acknowledge receipt of a business document.
Acceptance Acknowledgement (Non-substantive)	*timeToAcknowledgeAcceptance*
	The time a responding or requesting role has to non-substantively acknowledge business acceptance of a business document.

Substantive Response	*timeToPerform*
	The maximum amount of time between the time at which the resquest is sent and the substantive response is sent.

Note that the Acceptance Acknowledgement signal is often called the “non-substantive” response to the request.

A timeout parameter must be specified whenever a requesting or responding partner expects signals in return to the business document request or response. A requesting partner must not remain in an infinite wait state.

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

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

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

The total time allowed for a business transaction activity to complete is therefore, timeToPerform plus the timeToAcknowledgeReceipt on the response, and the timeToAcknowledgeAcceptance on the response. Additionaly, timeToPerform must be greated than the sum of timeAcknowledgeReceipt and timeToAcknowledge Acceptance and the request.

5.14.2 Protocol Exceptions

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

5.14.2.1 Receipt Acknowledgement Exception

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

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

- Syntax exceptions. There is invalid punctuation, vocabulary or grammar in the business document or business signal.
- Authorization exceptions. Roles are not authorized to participate in the business transaction activity. Note that this exception can only be identified by the receiving BSI.
- Signature exceptions. Business documents are not signed for non-repudiation when required.
- Sequence exceptions. The order or type of a business document or business signal is incorrect.

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

5.14.2.2 Acceptance Acknowledgement Exceptions

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

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

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

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

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

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

5.14.2.3 BSI compliance

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

IsAuthorizationRequired

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

IsNonRepudiationRequired

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

isNonRepudiationOfReceiptRequired.

Both partners agree to mutually verify receipt of a requesting business document and that the receipt must be non-repudiable. A requesting partner must initiate a notification of failure business transaction business (possibly revoking a contractual offer) if a responding partner has not properly delivered signed their receipt. For a further discussion of nonrepudiation of receipt, see also the ebXML E-Commerce and Simple Negotiation Patterns.

Non-repudiation of receipt provides the data for the following audit controls.
Verify responding role identity (authenticate) – Verify the identity of the responding role (individual or organization) that received the requesting business document.
Verify content integrity – Verify the integrity of the original content of the business document request.

isPositiveResponse

An expression whose evaluation results in TRUE or FALSE. If TRUE this DocumentEnvelope is intended as a positive response to the request. If isPositiveResponse = FALSE, the

2086	business transaction activity ends in business failure mode.
2087	The value for this parameter supplied for a DocumentEnvelope
2088	is an assertion by the sender of the DocumentEnvelope
2089	regarding its intent for the transaction to which it relates, but
2090	does not bind the recipient, or override the computation of
2091	transactional success or failure.

5.14.3 Computation of the status of a Business Transaction Activity

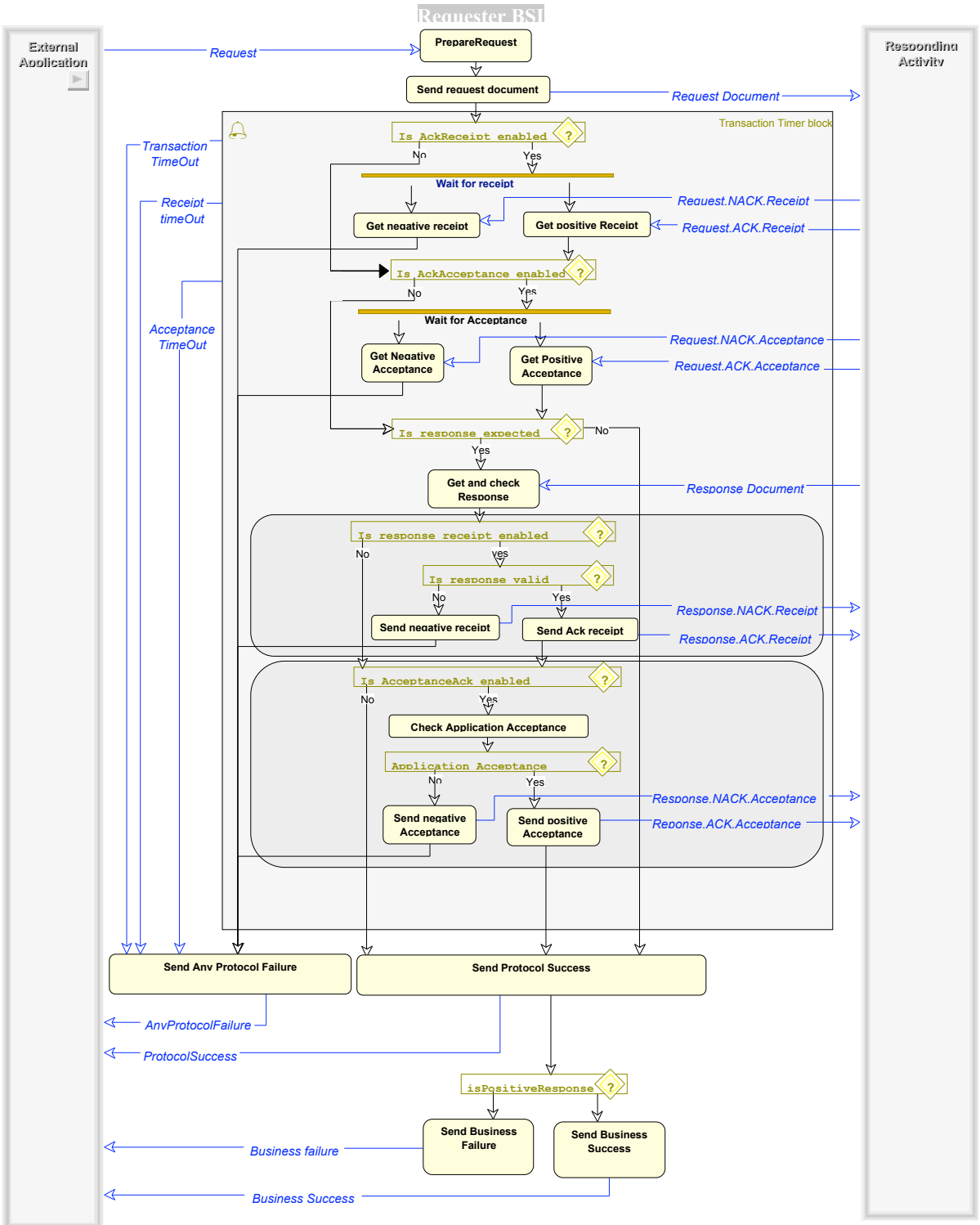


Figure 17. Computation of the Status of a Business Transaction Activity

Figure 17 represent the computation of the success or failure of a business transaction activity based on the different possible scenarios.

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

- ProtocolSuccess
- AnyProtocolFailure
  - RequestReceiptFailure
  - RequestAcceptanceFailure
  - ResponseReceiptFailure
  - ResponseAcceptanceFailure
  - SignalTimeout
  - ResponseTimeout
- BusinessSuccess (isPositiveResponse=true or no isPositiveResponse attribute)
- BusinessFailure(isPositiveResponse=false)
- Success (both protocol and business success)
- Failure (AnyProtocolFailure or BusinessFailure).

This figure does not represents the retryCount semantics.

BusinessFailure assumes that the transaction was successful from a "protocol" perspective, meaning that the state between the two parties could be effectively synchronized. However, the intent of the response was negative with respect to the request. As we mentioned earlier, this is an optional qualification of the response, agreed upon at design time, and some messages may not be qualifiable, i.e. they are neither positive or negative. The way business document specifications are designed allows to define two "logicial" documents from the same physical document and a condition expression evaluated at run-time by the BSI. If the condition is true and isPositiveResponse = false, then the transaction ends in business failure based on the business document content. Of course entire documents can be directly associated with isPositiveResponse=false, not just when they contain a particular field value.

It is required that each business transaction activity be designed such that there is at a minimum two transitions from the business transaction activity, one with a conditionGuard with a Success value, the other one with a Failure value, even if in case of failure the transitions goes to the failure state of the collaboration.

5.15 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.

5.16 Where the ebXML Business Process Specification Schema May Be Implemented

The ebXML *Business Process Specification Schema* should be used wherever software is being specified to perform a role in an ebXML business collaboration. Specifically, the ebXML *Business Process Specification*

Schema is intended to provide the business process and document specification for the formation of ebXML trading partner Collaboration Protocol Profiles and Agreements.

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

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

As of the current version, an ebXML compliant BSI is not requested that BSI be able to support multi-party collaboration. The current specification does not support the notions of context and correlation.

5.17 Guidelines for Business Service Interface Interoperability

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

Another important point for interoperability is that the context of a binary collaboration is limited to the document flows that are received or sent by the BSI. The BSI do not need to query information in other systems, internal or external to calculate the result of condition expressions.

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

5.18 Collaboration and transaction well-formedness rules

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

Business Transaction

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

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

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

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

[4] The time to perform a transaction cannot be null unless it is specified to be request without a response.

[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.

BusinessActivity

[7] Completion states must be defined on mutually exclusive paths guarantying that only one of the completion state will be reached.

[8] A BusinessActivity may have any number of incoming transition but only one output transition. Either a Fork or Decision business states must be used to logically specify more than one outgouing transition.

Business Collaboration

[9] There must be at most one Start business state in a binary collaboration defintion.

[10] There must be at least one Completion state in a binary collaboration definition

[11] A Role cannot perform both roles of the same business transaction activity.

[12] The two roles associated with a business collaboration must be different

6 ebXML Business Process Specification Schema –

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

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

6.1 Documentation for the Schema

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

targetNamespace: **http://www.ebxml.org/2003/1.1/BusinessProcess**

Elements

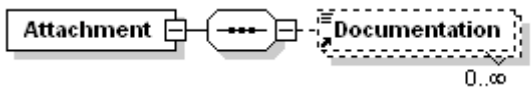
Attachment
AttributeSubstitution
BinaryCollaboration
BusinessDocument
BusinessPartnerRole
BusinessTransaction
BusinessTransactionActivity
CollaborationActivity
ConditionExpression
Decision
Documentation
DocumentEnvelope
DocumentSubstitution
Failure
Fork
Include
Join
MultiPartyCollaboration
Namespace
Namespaces
Package
Performs
ProcessSpecification
RequestingBusinessActivity
RespondingBusinessActivity
Start
SubstitutionSet
Success
Transition

Simple types

GUID
GUIDREF

6.1.1 element Attachment

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description An optional attachment to a BusinessDocument in a DocumentEnvelope.

Recommendation: Either use businessDocument +businessDocumentIDREF attributes OR use specification +mimeType attributes.

children **Documentation**

used by element **DocumentEnvelope**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the attachment.
nameID	GUID			GUIID version of name
businessDocument	xsd:string	required		A BusinessDocument can define an Attachment's type it is not of a defined Business Document, the mime typ and specification attribute will be the only indication of type.
businessDocumentIDREF	GUIDREF			The GUIIDREF version of businessDocument
specification	xsd:anyURI			A reference to an external source of description of this attachment.
mimeType	xsd:string	optional		Defines the valid MIME (Multipurpose Internet Mail Extensions) type of this Attachment. Example: 'application/pdf'
isAuthenticated	xsd:NMTOKEN			There is a digital certificate associated with the docume entity. This provides proof of the signer's identity.(See also section on Document Security)
isConfidential	xsd:NMTOKEN			The information entity is encrypted so that unauthorize parties cannot view the information(See also section or Document Security)
isTamperDetectable	xsd:NMTOKEN			The information entity has an encrypted message diges that can be used to check if the message has been tampered with. This requires a digital signature (sende digital certificate and encrypted message digest) associated with the document entity.(See also section Document Security)

source

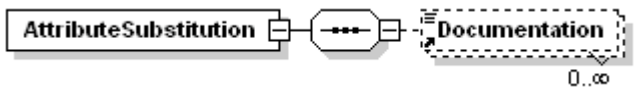
```
<xsd:element name="Attachment">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="businessDocument" type="xsd:string" use="required"/>
  <xsd:attribute name="businessDocumentIDREF"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

2250

2251

2252 # 6.1.2 element AttributeSubstitution

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description Attribute Substitution specifies that an attribute value should be used in place of some attribute value in an existing proces specification.

children **Documentation**

used by element **SubstitutionSet**

attributes

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

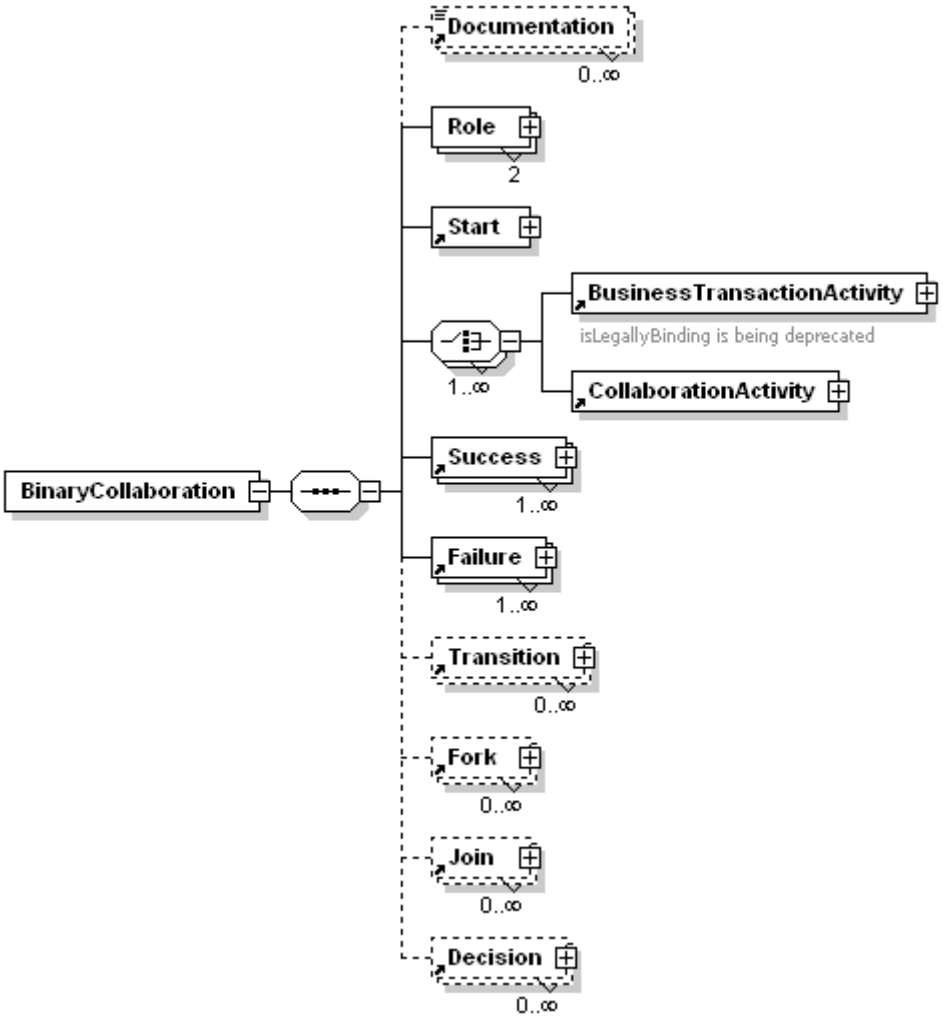
source

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

2253

2254 # 6.1.3 element BinaryCollaboration

Diagram



Namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description Binary Collaboration defines a protocol of interaction between two roles. One must be the initiating role, and one the responding role. Binary Collaboration is a choreographed set of states among collaboration roles. The activities of performing business transactions or other collaborations are a kind of state. Binary Collaboration choreographs one or mo business transaction activities between two roles. Binary Collaboration is not an atomic transaction. A binary collaboration may be used within another binary collaboration via a collaboration activity

Children **Documentation** **Role** **Start** **BusinessTransactionActivity** **CollaborationActivity** **Success** **Failure** **Transition** **Fork** **Joi** **Decision**

used by elements **Package** **ProcessSpecification**

Attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the attachment.
nameID	GUID			GUIID version of name
pattern	xsd:anyURI			The optional reference to a pattern that this binary collaboration is based on
beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaborati to commence
endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaborati to conclude
preCondition	xsd:string			A description of a state external to this collaborati that is required before this collaboration can commence
postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as result of the execution of this collaboration

timeToPerform	xsd:duration			The period of time, starting upon initiation of the fi activity, within which this entire collaboration mus conclude
initiatingRoleIDREF	GUIDREF	optional		Reference to the role that initiates the collaboratic Note that this just needs to be a logical reference, not an absolute value in case it could only be identified at run-time
isInnerCollaboration	xsd:boolean		false	Indicate whether or not this collaboration definitic can only be used within a collaboration activity (a sub collaboration) or initiated directly by a party.

Source

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

6.1.4 element BinaryCollaboration/Role

Diagram

Namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Type **RoleType**

Description Specifies the role name of a binary collaboration definition

children **Documentation**

attributes

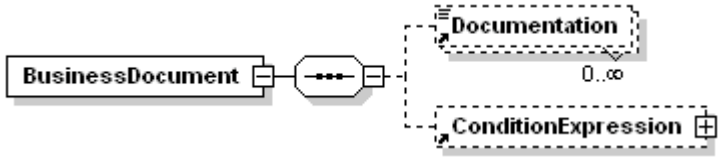
Name	Type	Use	Annotation
name	xsd:string	required	The name of the role
nameID	GUID	required	A GUID associated to the role. Note that the nameID must unique for all binary collaboration regardless if they reuse th same role name.

source

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

6.1.5 element BusinessDocument

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description BusinessDocument is a generic name of a document. A BusinessDocument may have one Condition Expression. This determines whether this is a valid business document for its envelope

determines whether this is a valid business document for its envelope

children **Documentation ConditionExpression**

used by
elements **Package ProcessSpecification**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		The logical name of the Business Document
nameID	GUID			A GUID associated with this document definitic
specificationLocation	xsd:anyURI			Reference to an external source of the schema definition. This defines the absolute path including the element name within the schema definition that defines the type of this documen
specificationID	xsd:anyURI			Absolute reference to the schema definition. T defines a unique identifier including the elemer id within the schema definition that defines the type of this document. Use either specificationLocation or specificationID
namespacePrefixes	xsd:NMTOKENS			Specifies a series of references to Namespace elements which are used by the schema definition if applicable.

source

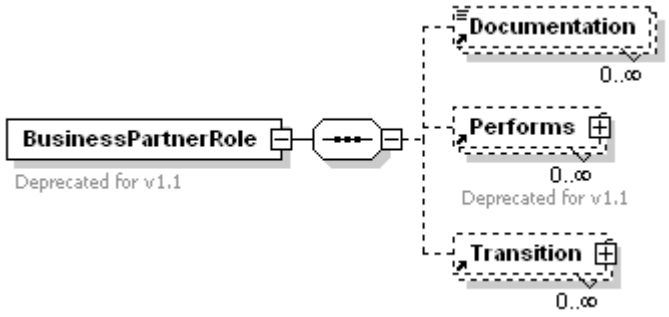
```
<xsd:element name="BusinessDocument">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ConditionExpression" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="specificationLocation"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

2259

2260

2261 # 6.1.6 element BusinessPartnerRole

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	A BusinessPartnerRole is the role played by a business partner in a MultiPartyCollaboration. A BusinessPartnerRole performs at most one Role in each of the Binary Collaborations that make up the Multiparty Collaboration. Wellformedness Rule: A partner must not perform both roles in a given business activity

children **Documentation** **Performs** **Transition**

used by element **MultiPartyCollaboration**

attributes

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

annotation documentation Deprecated for v1.1

source

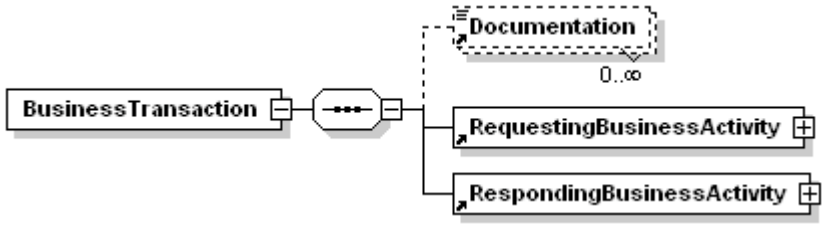
```
<xsd:element name="BusinessPartnerRole">
 <xsd:annotation>
  <xsd:documentation source="BPSS 1.1">Deprecated for v1.1</xsd:documentation>
 </xsd:annotation>
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Performs" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Transition" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
 </xsd:complexType>
</xsd:element>
```

2262

2263

2264 # 6.1.7 element BusinessTransaction

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

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

children Documentation RequestingBusinessActivity RespondingBusinessActivity

used by

elements Package ProcessSpecification

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the Business Transaction.
nameID	GUID			The GUID version of the name
pattern	xsd:anyURI			The optional reference to a pattern that this transaction is based on the UN/CEFACT UMM specification
isGuaranteedDeliveryRequired	xsd:boolean		false	Both partners must agree to use a transport that guarantees delivery

source

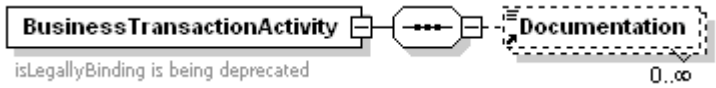
```
<xsd:element name="BusinessTransaction">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="RequestingBusinessActivity"/>
   <xsd:element ref="RespondingBusinessActivity"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>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

2265

2266

2267 # 6.1.8 element BusinessTransactionActivity

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

type extension of **BusinessActivity**

Description	A business transaction activity defines the use of a business transaction within a binary collaboration. A business transacti activity is a business activity that executes a specified business transaction. More than one instance of the same business transaction activity can be open at one time if the isConcurrent property is true. A Role may not be both the requestor and the responder in a business transaction.

children **Documentation**

used by element **BinaryCollaboration**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the activity uniquely within th binary collaboration
nameID	GUID			The GUID version of the name
fromRole	xsd:string	required		The name of the initiating role in Business Transaction Activity. This must match one of the roles of the binary collaboration and will become t requestor in the BusinessTransaction performed I this activity
fromRoleIDREF	GUIDREF			The GUIDREF version of fromRole
toRole	xsd:string	required		The name of the responding role in Business Transaction Activity. This must match one of the roles in the binary collaboration and will become t responder in the BusinessTransaction performed this activity
toRoleIDREF	GUIDREF			The GUIDREF version of toRole
beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaborati to commence
endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaborati to conclude
preCondition	xsd:string			A description of a state external to this collaborati that is required before this collaboration can commence
postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as result of the execution of this collaboration
businessTransaction	xsd:string	required		A reference, by name to the Business Transactior performed by this Business Transaction Activity
businessTransactionIDREF	GUIDREF			A GUIDREF reference to the Business Transactic GUID
isConcurrent	xsd:boolean		true	If the BusinessTransactionActivity is concurrent th more than one instance of the associated BusinessTransaction can be open the same time part of the execution of this Business Transaction Activity regardless of the Binary Collaboration instance
isLegallyBinding	xsd:boolean		true	Defines whether the Business Transaction performed by this activity is intended by the tradin parties to be binding. Default value is True.
timeToPerform	xsd:duration			The period of time, starting upon the sending of th request, within which the response will be sent ba

annotation isLegallyBinding is being deprecated

source

```
<xsd:element name="BusinessTransactionActivity">
 <xsd:annotation>
  <xsd:documentation>isLegallyBinding is being deprecated</xsd:documentation>
 </xsd:annotation>
 <xsd:complexType>
  <xsd:complexContent>
   <xsd:extension base="BusinessActivity">
    <xsd:sequence>
     <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
```

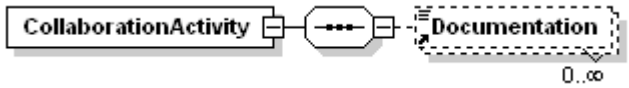
```
        <xsd:attribute name="businessTransaction" type="xsd:string" use="required"/>
        <xsd:attribute name="businessTransactionIDREF" type="GUIDREF"/>
        <xsd:attribute name="isConcurrent" type="xsd:boolean" default="true"/>
        <xsd:attribute name="isLegallyBinding" type="xsd:boolean" default="true">
          <xsd:annotation>
            <xsd:documentation source="BPSS 1.1">Deprecated for v1.1</xsd:documentation>
          </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="timeToPerform" type="xsd:duration"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

2268

2269

2270 # 6.1.9 element CollaborationActivity

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

type extension of **BusinessActivity**

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

children **Documentation**

used by element **BinaryCollaboration**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the activity uniquely within th binary collaboration
nameID	GUID			The GUID version of the name
fromRole	xsd:string	required		The name of the initiating role in Business Transaction Activity. This must match one of the roles of the binary collaboration and will become t requestor in the BusinessTransaction performed I this activity
fromRoleIDREF	GUIDREF			The GUIDREF version of fromRole
toRole	xsd:string	required		The name of the responding role in Business Transaction Activity. This must match one of the roles in the binary collaboration and will become t responder in the BusinessTransaction performed this activity
toRoleIDREF	GUIDREF			The GUIDREF version of toRole
beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaborati to commence
endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaborati to conclude
preCondition	xsd:string			A description of a state external to this collaborati that is required before this collaboration can commence
postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as result of the execution of this collaboration
binaryCollaboration	xsd:string	required		A reference, by name to the Binary Collaboration performed by this Collaboration Activity
binaryCollaborationIDREF	GUIDREF			A GUIDREF reference to the Binary Collaboratior definition GUID

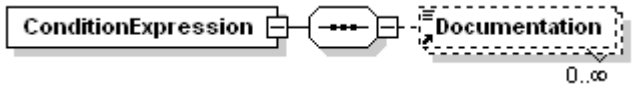
source

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

2271
2272

2273 # 6.1.10 element ConditionExpression

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	Condition Expression is an expression that can be evaluated to TRUE or FALSE.

children **Documentation**

used by
elements **BusinessDocument Decision Failure Success Transition**

attributes

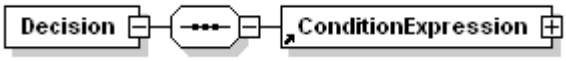
Name	Type	Use	Default	Annotation
expressionLanguage	xsd:string	required		The language of the expression, e.g. XPATH 1.0 or DocumentEnvelopeNotation
expression	xsd:string	required		An expression whose evaluation results in TRUE or FALSE. For a transition, this determines whether th transition should happen or not. For a business document, this determines whether this is a valid business document for its envelope. The expression can refer to the name or content of the most recent DocumentEnvelope or content of documents within i
prefix	xsd:string	optional		Namespace prefix used by the XPATH expression

source

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

6.1.11 element Decision

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description A pseudo-state that matches the semantics of the Decision element of the UML activity diagram. This element outgoing transition are by definition mutually exclusive.

children **ConditionExpression**

used by
element **BinaryCollaboration**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		The name of the Decision element
nameID	GUID			A GUID version of the name

source

```
<xsd:element name="Decision">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="ConditionExpression"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
 </xsd:complexType>
</xsd:element>
```

6.1.12 element Documentation

diagram

namespace http://www.ebxml.org/2003/1.1/BusinessProcess

type extension of **xsd:string**

Description	Defines user documentation for any element. Must be the first element of its container. Documentation can be either inline PCDATA and/or a URI to where more complete documentation is to be found

used by
elements **Attachment AttributeSubstitution BinaryCollaboration BusinessDocument BusinessPartnerR BusinessTransaction BusinessTransactionActivity CollaborationActivity ConditionExpressio DocumentEnvelope DocumentSubstitution Failure Fork Include Join MultiPartyCollaboration Package Performs ProcessSpecification Start SubstitutionSet Success Transition**
complexTypes **BusinessAction RoleType**

attributes	Name	Type	Use	Default	Fixed	Annotation
	uri	xsd:anyURI				

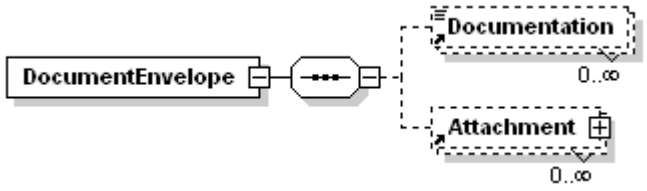
source

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

2280

6.1.13 element DocumentEnvelope

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	A DocumentEnvelope is what conveys business information between the two roles in a business transaction. One DocumentEnvelope conveys the request from the requesting role to the responding role, and another DocumentEnvelope conveys the response (if any) from the responding role back to the requesting role. A documentEnvelop contains exactly o primary Business document. It contains an optional set of attachments related to primary document. Wellformedness Rules: A Document Envelope is associated with exactly one requesting and one responding activity. IsPositiveResponse is not a relevant parameter on a DocumentEnvelope sent by a requesting activity

children **Documentation** **Attachment**

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

attributes

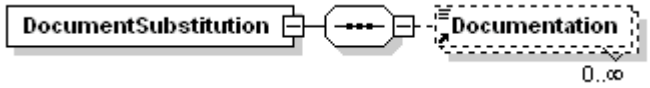
Name	Type	Use	Annotation
name	xsd:string		Defines Name of the DocumentEnvelope
nameID	GUID		Defines GUID of the DocumentEnvelope
businessDocument	xsd:string	required	The name of the business document.
businessDocumentIDREF	GUIDREF		The GUIREF version of businessDocument
isPositiveResponse	xsd:boolean		TRUE or FALSE. If TRUE this DocumentEnvelope is intend as a positive response to the request. The value for this parameter is used to evaluate a Business Success or Failur of the corresponding Business Transaction.
isAuthenticated	xsd:NMTOKEN		There is a digital certificate associated with the document entity. This provides proof of the signer's identity.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".
isConfidential	xsd:NMTOKEN		The information entity is encrypted so that unauthorized parties cannot view the information.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".
isTamperDetectable	xsd:NMTOKEN		The information entity has an encrypted message digest tha can be used to check if the message has been tampered wi This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the docume entity.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "a least value".

source

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

6.1.14 element DocumentSubstitution

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	DocumentSubstitution specifies a document that should be used in place of a document in an existing process specificatio

children **Documentation**

used by element **SubstitutionSet**

attributes

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

source

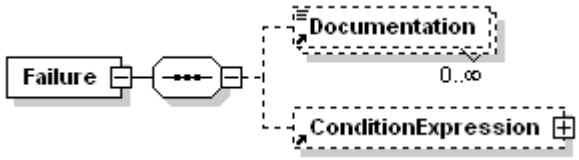
```
<xsd:element name="DocumentSubstitution">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="originalBusinessDocument" type="xsd:string" use="required"/>
  <xsd:attribute name="originalBusinessDocumentID"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

2285

2286

2287 ## 6.1.15 element Failure

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	Defines the unsuccessful conclusion of a binary collaboration as a transition from an activity. Wellformedness Rules: Every Binary Collaboration should have at least one failure.

children **Documentation** **ConditionExpression**

used by element **BinaryCollaboration**

attributes

Name	Type	Use	Default	Annotation
nameID	GUID			Defines GUID of the Failure
fromBusinessState	xsd:string	required		The name of the activity from which this indicates transition to unsuccessful conclusion of the BusinessTransaction or BinaryCollaboration
fromBusinessStateIDREF	GUIDREF			The GUIDREF version of fromBusinessState
conditionGuard	xsd:NMTOKEN			The condition that guards this transition

source

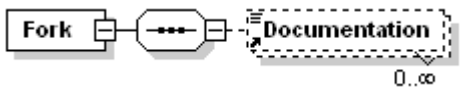
```
<xsd:element name="Failure">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ConditionExpression" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="nameID" type="GUID"/>
  <xsd:attribute name="fromBusinessState"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

2288

2289

2290 ## 6.1.16 element Fork

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	A Fork is a state with one inbound transition and multiple outbound transitions. All activities pointed to by the outbound transitions are assumed to happen in parallel or exclusive or.

children **Documentation**

used by element **BinaryCollaboration**

attributes

Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the Fork state
nameID	GUID			The GUID version of name
type	xsd:NMTOKEN	optional	OR	All activities will run in parallel. XOR: Only one the possible activities will run.
timeToPerform	xsd:duration	optional		timeToPerform attribute on the Fork element m be used to specify that the business activities between the Fork and the Join shall be execute within the specified duration otherwise, the stat of the collaboration will automatically advance the join.

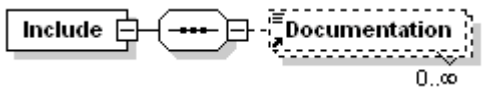
source

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

2291

6.1.17 element Include

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

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

children **Documentation**

used by elements **Package** **ProcessSpecification**

attributes

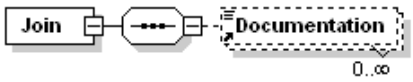
Name	Type	Use	Default	Annotation
name	xsd:string	required		Name of the included specification
nameID	GUID	required		Unique identifier of the included specification
uri	xsd:anyURI	required		URI of the included specification
version	xsd:string	required		Version of the included specification

source

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

6.1.18 element Join

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	A business state where an activity is waiting for the completion of one or more other activities. Defines the point where previously forked activities join up again.

children **Documentation**

used by element **BinaryCollaboration**

attributes

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

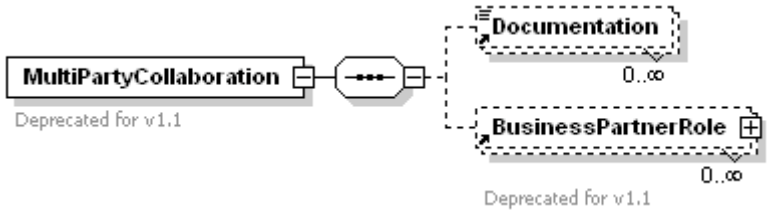
source

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

2297

2298 ## 6.1.19 element MultiPartyCollaboration

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	A Multiparty Collaboration is a synthesis of Binary Collaborations. A Multiparty Collaboration consists of a number of Business Partner Roles each playing roles in binary collaborations with each other. Wellformedness Rules: All multiparty collaborations must be synthesized from binary collaborations

children **Documentation** **BusinessPartnerRole**

used by
elements **Package** **ProcessSpecification**

attributes

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

annotation
documentation Deprecated for v1.1

source

```
<xsd:element name="MultiPartyCollaboration">
 <xsd:annotation>
  <xsd:documentation source="BPSS 1.1">Deprecated for v1.1</xsd:documentation>
 </xsd:annotation>
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="BusinessPartnerRole" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
 </xsd:complexType>
</xsd:element>
```

2299

2300

2301 ## 6.1.20 element Namespace

diagram

namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description A Namespace definition. A Namespace may also have possible (optional) alternative namespaces

children **Namespace**

used by elements **Namespace** **Namespaces**

attributes

Name	Type	Use	Default	Annotation
URI	xsd:anyURI	required		URI of the namespace definition
prefix	xsd:NMTOKEN	required		Namespace prefix
nameID	GUID	required		The GUID of the Namespace definition

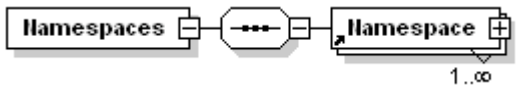
source

```
<xsd:element name="Namespace">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:annotation>
    <xsd:documentation>alternative namespaces</xsd:documentation>
   </xsd:annotation>
   <xsd:element ref="Namespace" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="prefix" type="xsd:NMTOKEN" use="required"/>
  <xsd:attribute name="nameID" type="GUID" use="required"/>
 </xsd:complexType>
</xsd:element>
```

2302
2303

2304 ## 6.1.21 element Namespaces

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description Container element for the Namespace definitions.

children **Namespace**

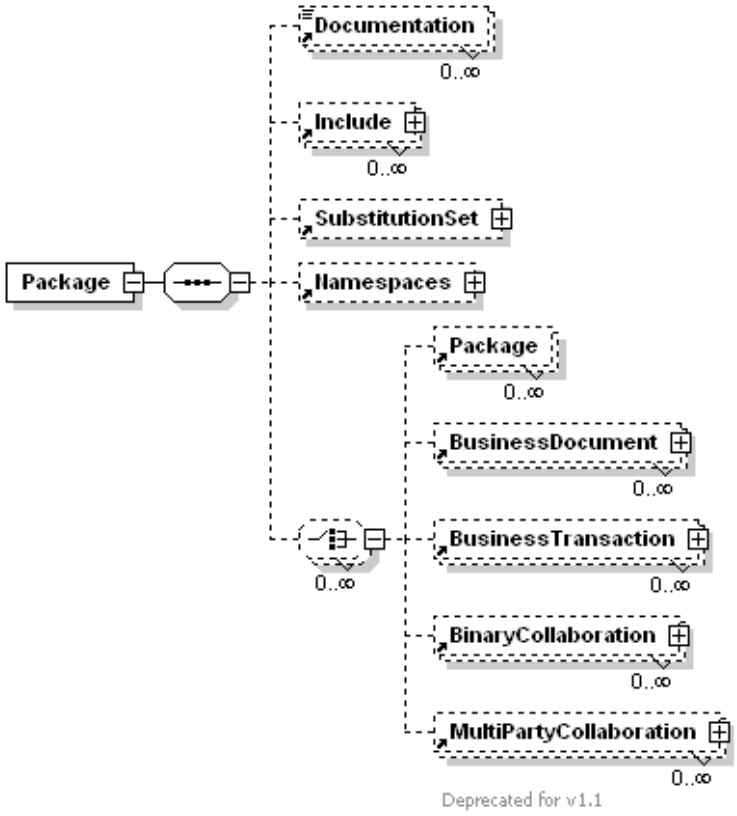
used by elements **Package** **ProcessSpecification**

source

```
<xsd:element name="Namespaces">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Namespace" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

6.1.22 element Package

diagram



namespace	http://www.ebxml.org/2003/1.1/BusinessProcess
Description	Defines a hierarchical name scope containing reusable elements.
children	Documentation Include SubstitutionSet Namespaces Package BusinessDocument BusinessTransaction BinaryCollaboration MultiPartyCollaboration
used by	elements Package ProcessSpecification

attributes

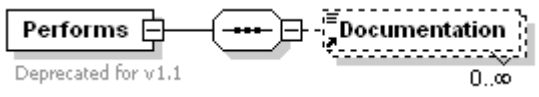
Name	Type	Use	Default	Annotation
name	xsd:string	required		Name of the package
nameID	GUID			GUID version of name

source

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

6.1.23 element Performs

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	Performs is an explicit modeling of the relationship between a BusinessPartnerRole and the Roles it plays. This specifies t use of an Authorized Role within a multiparty collaboration. Wellformedness Rules: For every Performs performing a Role there must be a Performs that performs the opposing Role, otherwise the MultiParty Collaboration is not complete.

children **Documentation**

used by element **BusinessPartnerRole**

attributes

Name	Type	Use	Default	Annotation
nameID	GUID			GUID of the Performs element
role	xsd:string	required		The Role that will be performed by the Business PartnerRole, qualified with the name of the BinaryCollaboration
roleIDREF	GUIDREF	required		GUIDREF version of Role

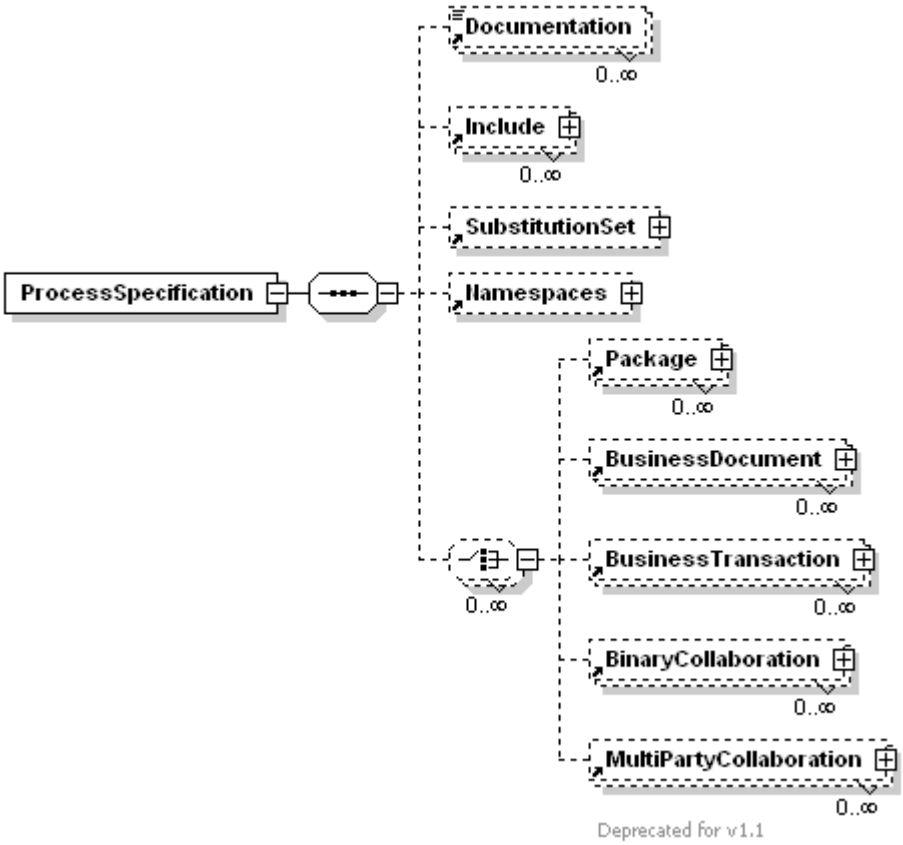
annotation documentation Deprecated for v1.1

source

```
<xsd:element name="Performs">
 <xsd:annotation>
  <xsd:documentation source="BPSS 1.1">Deprecated for v1.1</xsd:documentation>
 </xsd:annotation>
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="nameID" type="GUID"/>
  <xsd:attribute name="role" type="xsd:string" use="required"/>
  <xsd:attribute name="roleIDREF" type="GUIDREF" use="required"/>
 </xsd:complexType>
</xsd:element>
```

6.1.24 element ProcessSpecification

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

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

children **Documentation Include SubstitutionSet Namespaces Package BusinessDocument BusinessTransaction BinaryCollaboration MultiPartyCollaboration**

attributes

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

source

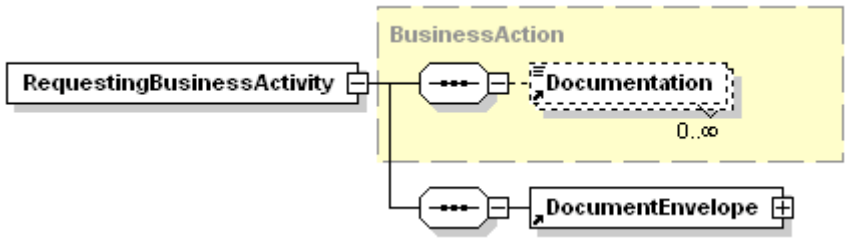
```
<xsd:element name="ProcessSpecification">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Include" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="SubstitutionSet" minOccurs="0"/>
   <xsd:element ref="Namespaces" minOccurs="0"/>
   <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="Package" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BusinessDocument" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BusinessTransaction" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="BinaryCollaboration" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="MultiPartyCollaboration" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="nameID" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="version" type="xsd:string" use="required"/>
 </xsd:complexType>
 <xsd:unique name="ProcessSpecification-ID">
  <xsd:selector xpath="."/>
  <xsd:field xpath="nameID"/>
 </xsd:unique>
</xsd:element>
```

2311

2312

2313 # 6.1.25 element RequestingBusinessActivity

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

type extension of **BusinessAction**

Description	A RequestingBusinessActivity is a Business Action that is performed by the requesting role within a Business Transaction. specifies the Document Envelope which will carry the request.

children **Documentation** **DocumentEnvelope**

used by element **BusinessTransaction**

attributes

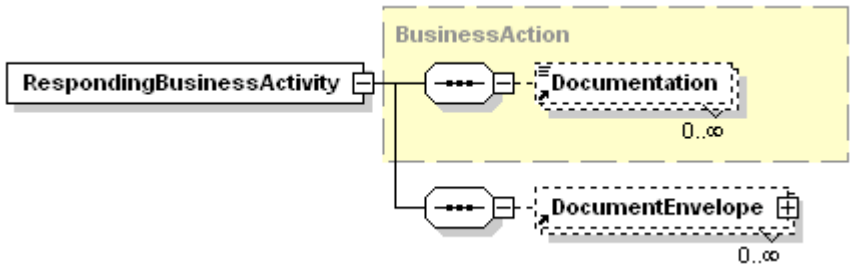
Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the RequestingBusinessTransactic
nameID	GUID			The GUID version of name
isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originator agai a list of authorized originators.This parameter is specifi on the sending side. (See also section on action securit
isIntelligibleCheckRequired	xsd:boolean		false	Receiving party must check that a requesting documen not garbled (unreadable, unintelligible) before sending acknowledgement of receipt This parameter is specifie on the sending side.(See also section on core transacti semantics)
isNonRepudiationRequired	xsd:boolean		false	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt.This parameter is specified on the sending side.(See also section on nonrepuditation)
isNonRepudiationReceiptRequired	xsd:boolean		false	Requires the sending parties to save copies of the transacted documents before sending them(See also section on nonrepuditation)
timeToAcknowledgeReceipt	xsd:duration			The time a responding role has to non-substantively acknowledge business acceptance of a business document.This parameter is specified on the requesting side.(See also section on core transaction semantics)
timeToAcknowledgeAcceptance	xsd:duration			The time the receiving party has to acknowledge receip of a business document. This parameter is specified on the sending side.(See also section on core transaction semantics)
retryCount	xsd:int			The BSI must retry to send a request n number of times in case no signals are returned by the responding activi

source

```
<xsd:element name="RequestingBusinessActivity">
 <xsd:complexType>
  <xsd:complexContent>
   <xsd:extension base="BusinessAction">
    <xsd:sequence>
     <xsd:element ref="DocumentEnvelope"/>
    </xsd:sequence>
    <xsd:attribute name="retryCount" type="xsd:int"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
```

6.1.26 element RespondingBusinessActivity

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

type extension of **BusinessAction**

Description	A RespondingBusinessActivity is a Business Action that is performed by the responding role within a Business Transactio It specifies the Document Envelope which will carry the response.There may be multiple possible response Document Envelopes defined, but only one of them will be sent during an actual transaction instance.

children **Documentation** **DocumentEnvelope**

used by element **BusinessTransaction**

attributes

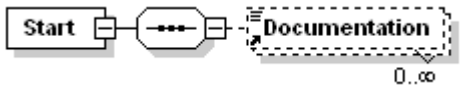
Name	Type	Use	Default	Annotation
name	xsd:string	required		Defines the name of the RespondingBusinessTransaction
nameID	GUID			The GUID version of name
isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originato against a list of authorized originators.This parameter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

source

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

6.1.27 element Start

diagram



namespace	http://www.ebxml.org/2003/1.1/BusinessProcess
Description	The starting state for an Binary Collaboration. A Binary Collaboration should have only one starting activity.
children	**Documentation**
used by	element **BinaryCollaboration**

attributes

Name	Type	Use	Default	Annotation
toBusinessState	xsd:string	required		The name of an activity which an allowable starting point for this for BinaryCollaboration
toBusinessStateIDREF	GUIDREF			The GUIIDREF version of toBusinessState
nameID	GUIDREF			The GUID of the Start element

source

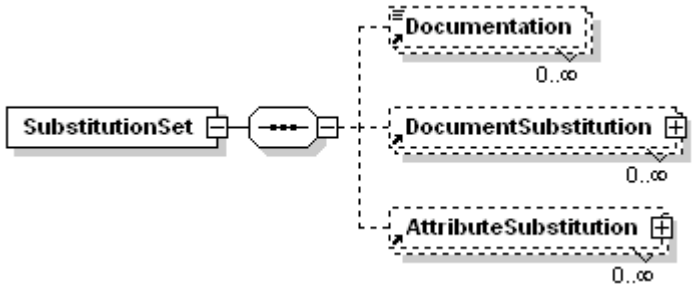
```
<xsd:element name="Start">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="toBusinessState" type="xsd:string" use="required"/>
  <xsd:attribute name="toBusinessStateIDREF" type="GUIDREF"/>
  <xsd:attribute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

2318

2319

2320 ## 6.1.28 element SubstitutionSet

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	A Substitution Set is a container for one or more AttributeSubstitution and/or DocumentSubstitution elements. The entire SubstitutionSet specifies document or attribute values that should be used in place of some documents and attribute value in an existing process specification.

children **Documentation** **DocumentSubstitution** **AttributeSubstitution**

used by elements **Package** **ProcessSpecification**

attributes

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

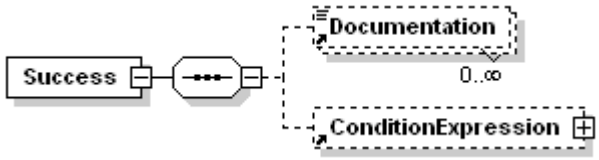
source

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

2321

6.1.29 element Success

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

Description	Defines the successful conclusion of a binary collaboration as a transition from an activity. Wellformedness Rules: Every activity Binary Collaboration should have at least one success.

children **Documentation** **ConditionExpression**

used by element **BinaryCollaboration**

attributes

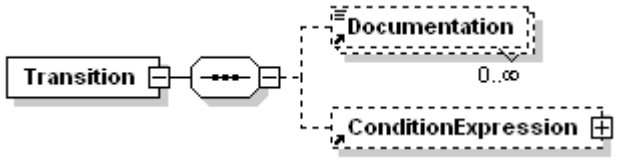
Name	Type	Use	Default	Annotation
nameID	GUID			Defines GUID of the Success element
fromBusinessState	xsd:string	required		The name of the activity from which this indicates transition to successful conclusion of the BusinessTransaction or BinaryCollaboration
fromBusinessStateIDREF	GUIDREF			The GUIDREF of the business state
conditionGuard	xsd:NMTOKEN			The condition that guards this transition

source

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

6.1.30 element Transition

diagram



namespace http://www.ebxml.org/2003/1.1/BusinessProcess

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

children **Documentation** **ConditionExpression**

used by elements **BinaryCollaboration** **BusinessPartnerRole**

attributes

Name	Type	Use	Default	Annotation
nameID	GUID			Defines GUID of the Transition
onInitiation	xsd:boolean		false	This specifies this is a nested BusinessTransactionActivity and that upon receipt of th request in the associated transaction a second activity i performed before returning to the transaction to send th response back to the original requestor
fromBusinessState	xsd:string	required		The name of the state transitioned from.
fromBusinessStateIDREF	GUIDREF	optional		The GUIDREF version of fromBusinessState
toBusinessState	xsd:string	required		The name of the state transitioned to
toBusinessStateIDREF	GUIDREF	optional		The GUIDREF version of toBusinessState
conditionGuard	xsd:NMTOKEN			A reference to the status of the previous transaction

source

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

6.1.31 simpleType GUID

namespace http://www.ebxml.org/2003/1.1/BusinessProcess

type **xsd:string**

Description All elements are required to have GUID (instead of xs:ID) because of the notion of includes and packages, which would le to invalid XML document if xs:ID and xs:IDREF were used.

used by attributes **DocumentEnvelope/@nameID** **Failure/@nameID** **Performs/@nameID** **Success/@nameID** **Transition/@nameID** **Namespace/@nameID** **RoleType/@nameID** **name/@nameID** **Include/@nameID**

source

```
<xsd:simpleType name="GUID">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

6.1.32 simpleType GUIDREF

namespace http://www.ebxml.org/2003/1.1/BusinessProcess

type **xsd:string**

Description: All elements are required to have GUID (instead of xs:ID) because of the notion of includes and packages, which would le to invalid XML document if xs:ID and xs:IDREF were used.

used by attributes: CollaborationActivity/@binaryCollaborationIDREF Attachment/@businessDocumentIDREF DocumentEnvelope/@businessDocumentIDREF BusinessTransactionActivity/@businessTransactionIDREF Failure/@fromBusinessStateIDREF Success/@fromBusinessStateIDREF Transition/@fromBusinessStateIDREF BusinessActivity/@fromRoleIDREF BinaryCollaboration/@initiatingRoleIDREF Start/@nameID DocumentSubstitution/@originalBusinessDocumentID Performs/@roleIDREF Start/@toBusinessStateIDREF Transition/@toBusinessStateIDREF BusinessActivity/@toRoleIDREF

source

```
<xsd:simpleType name="GUIDREF">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

6.2 XML to UML cross-reference

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

XML Element	UML Class
Attachment	Attachment
Role	AuthorizedRole
Binary Collaboration	Binary Collaboration
BusinessPartner Role	BusinessPartner Role
Business Transaction Activity	Business Transaction Activity
Business Transaction	Business Transaction
Responding BusinessActivity	Responding BusinessActivity
Requesting BusinessActivity	Requesting BusinessActivity
Collaboration Activity	Collaboration Activity
DocumentEnvelope	DocumentEnvelope
Documentation	None (Should be added)
ebXML Process Specification	(From Package model: ebXML Process Specification)
Failure	Failure
Include	(From Package model: Include)
MultiParty Collaboration	MultiParty Collaboration
Package	(From Package model: Package)
Performs	Performs
Schema	Schema
Decision	Decision
Fork	Fork
Start	Start
Success	Success
Join	Join
Transition	Transition
BusinessAction	BusinessAction
DocumentSecurity	DocumentSecurity

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

- BusinessState
- CompletionState

- BusinessActivity

6.3 Scoped Name Reference

The structure of ebXML Business Process Specification Schema encourages re-use. A BPSS instance can include another BPSS instance by reference.

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

Packages function as namespaces as per below.

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

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

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

Within an ebXML BPSS instance the following applies to naming:

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

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

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

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

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

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

```
<Performs Role='//Role[@name="buyer"]'/>
Relative paths are also allowed for example:
<BusinessTransactionActivity fromRole='../ Role[@name="buyer"]'
... />
```

6.4 Sample XML document against above Schema

Provided in Appendix A

7 Business signal structures

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

An Exception message would be sent in lieu of a ReceiptAcknowledgement signal or an AcceptanceAcknowledgment signal and would indicate a corresponding negative ReceiptAcknowledgement or negative AcceptanceAcknowledgement. On the other hand, sending a ReceiptAcknowledgment or AcceptanceAcknowledgement message as defined below would indicate a positive signal.

7.1.1 Signal Schema

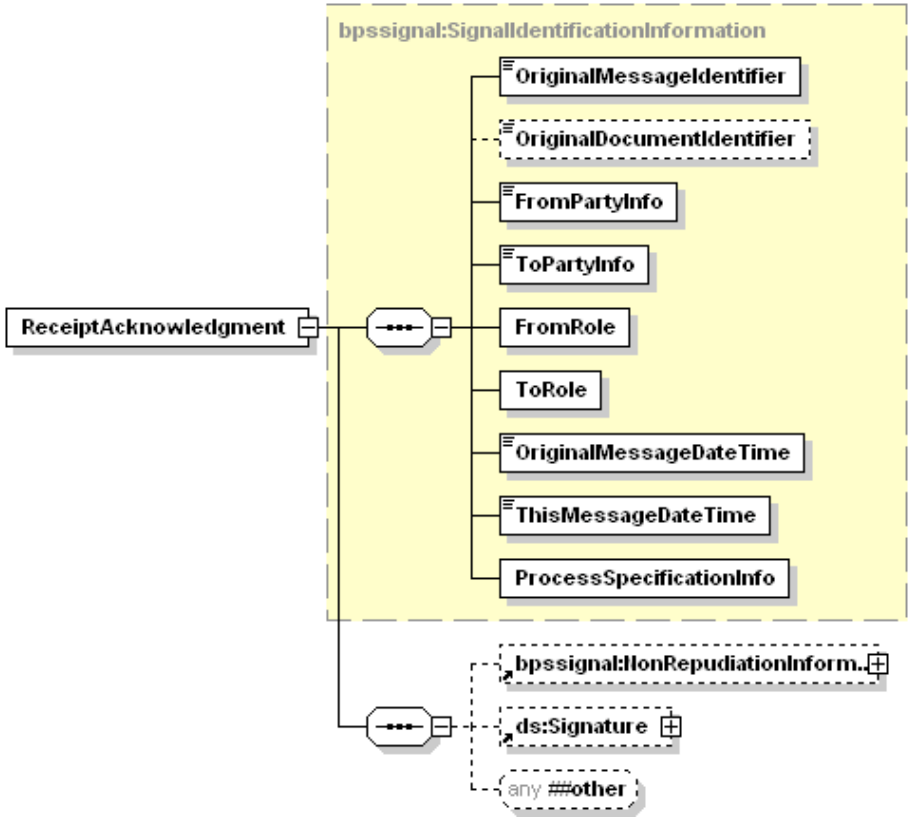
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- By Himagiri Mukkamala(himagiri@sybase.com) .
This schema has the element definitions for the signal messages used in the run time execution of BPSS-->
<xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="qualified" version="2.0">
    <xsd:import namespace="http://www.w3.org/1999/xlink" schemaLocation="http://www.oasis-
open.org/committees/ebxml-msg/schema/xlink.xsd"/>
    <xsd:annotation>
        <xsd:documentation>
The version of digital signature specification supported is identified using the namespace and schemalocation
denoted below      </xsd:documentation>
    </xsd:annotation>
    <xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
    <xsd:simpleType name="non-empty-string">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="PartyInfoType">
        <xsd:simpleContent>
            <xsd:extension base="non-empty-string">
                <xsd:attribute name="type" type="non-empty-string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="RoleType">
        <xsd:annotation>
            <xsd:documentation>
This type defines the structure for Role Definition.
             </xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="name" type="non-empty-string" use="required"/>
        <xsd:attributeGroup ref="xlink.grp"/>
    </xsd:complexType>
    <xsd:attributeGroup name="xlink.grp">
        <xsd:attribute ref="xlink:type" fixed="simple"/>
        <xsd:attribute ref="xlink:href" use="required"/>
    </xsd:attributeGroup>
```

```
    <xsd:complexType name="ProcessSpecificationType">
        <xsd:attribute name="version" type="non-empty-string"/>
        <xsd:attribute name="name" type="non-empty-string"/>
        <xsd:attributeGroup ref="xlink.grp"/>
        <xsd:attribute name="nameID" type="xsd:anyURI"/>
    </xsd:complexType>
    <xsd:complexType name="SignalIdentificationInformation">
        <xsd:sequence>
            <xsd:element name="OriginalMessageIdentifier" type="bpssignal:non-empty-string"/>
            <xsd:element name="OriginalDocumentIdentifier" type="bpssignal:non-empty-string"
minOccurs="0"/>
            <xsd:element name="FromPartyInfo" type="bpssignal:PartyInfoType"/>
            <xsd:element name="ToPartyInfo" type="bpssignal:PartyInfoType"/>
            <xsd:element name="FromRole" type="bpssignal:RoleType"/>
            <xsd:element name="ToRole" type="bpssignal:RoleType"/>
            <xsd:element name="OriginalMessageDateTime" type="xsd:dateTime"/>
            <xsd:element name="ThisMessageDateTime" type="xsd:dateTime"/>
            <xsd:element name="ProcessSpecificationInfo" type="bpssignal:ProcessSpecificationType"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="Exception">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="bpssignal:SignalIdentificationInformation">
                    <xsd:sequence>
                        <xsd:element name="ExceptionType">
                            <xsd:complexType>
                                <xsd:choice>
                                    <xsd:element name="ReceiptException">
                                        <xsd:simpleType>
                                            <xsd:restriction base="xsd:string">
                                                <xsd:enumeration value="Syntax"/>
                                                <xsd:enumeration value="Authorization"/>
                                                <xsd:enumeration value="Signature"/>
                                                <xsd:enumeration value="Sequence"/>
                                            </xsd:restriction>
                                        </xsd:simpleType>
                                    </xsd:element>
                                    <xsd:element name="AcceptanceException">
                                        <xsd:simpleType>
                                            <xsd:restriction base="xsd:string">
                                                <xsd:enumeration value="Business"/>
                                                <xsd:enumeration value="Performance"/>
                                            </xsd:restriction>
                                        </xsd:simpleType>
                                    </xsd:element>
                                    <xsd:element name="GeneralException">
                                        <xsd:simpleType>
                                            <xsd:restriction base="xsd:string"/>
                                        </xsd:simpleType>
                                    </xsd:element>
                                </xsd:choice>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="Reason" type="bpssignal:non-empty-string"/>
                        <xsd:element name="ExceptionMessage" type="bpssignal:non-empty-string"
                                     minOccurs="0"/>
                        <xsd:any namespace="##other" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ReceiptAcknowledgment">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="bpssignal:SignalIdentificationInformation">
                    <xsd:sequence>
                        <xsd:element ref="bpssignal:NonRepudiationInformation" minOccurs="0"/>
                        <xsd:element ref="ds:Signature" minOccurs="0"/>
                        <xsd:any namespace="##other" minOccurs="0"/>
```

```
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="NonRepudiationInformation">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="bpssignal:MessagePartNRInformation"  maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="MessagePartNRInformation">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element name="MessagePartIdentifier"  type="bpssignal:non-empty-string"/>
                <xsd:element ref="ds:Reference"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AcceptanceAcknowledgment">
        <xsd:annotation>
            <xsd:documentation>
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="bpssignal:SignalIdentificationInformation">
                    <xsd:sequence>
                        <xsd:element ref="ds:Signature"  minOccurs="0"/>
                        <xsd:any namespace="##other"  minOccurs="0"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

2588 7.1.2 ReceiptAcknowledgment Signal Schema

diagram



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

type extension of **bpssignal:SignalIdentificationInformation**

children **OriginalMessageIdentifier** **OriginalDocumentIdentifier** **FromPartyInfo** **ToPartyInfo** **FromRole** **ToRole** **OriginalMessageDateTime** **ThisMessageDateTime** **ProcessSpecificationInfo** **bpssignal:NonRepudiationInformatio** **ds:Signature**

annotation This defines the content structure for messages that need to send an ReceiptAcknowledgment signals as a business message to a trading partner. Please refer to BPSS document for detailed description of ReceiptAcknowledgment.
For description of first nine elements, refer to documentation on SignalIdentificationInformation.
ReceiptAcknowledgment signals can include non-repudiation information if requested in the process definition.
"NonRepudiationInformation" captures this data for each of the message parts that comprise the request message that was sent. Each "MessagePartNRInformation" describes non-repudiation information for a message part identified by "MessagePartIdentifier" using "Reference" described by XML Digital Signature Specification. Each part of the request message will have a corresponding "MessagePartNRInformation".
If necessary, digital signature can be computed for this signal message and included using "Signature" element from XM Digital Signature namespace.

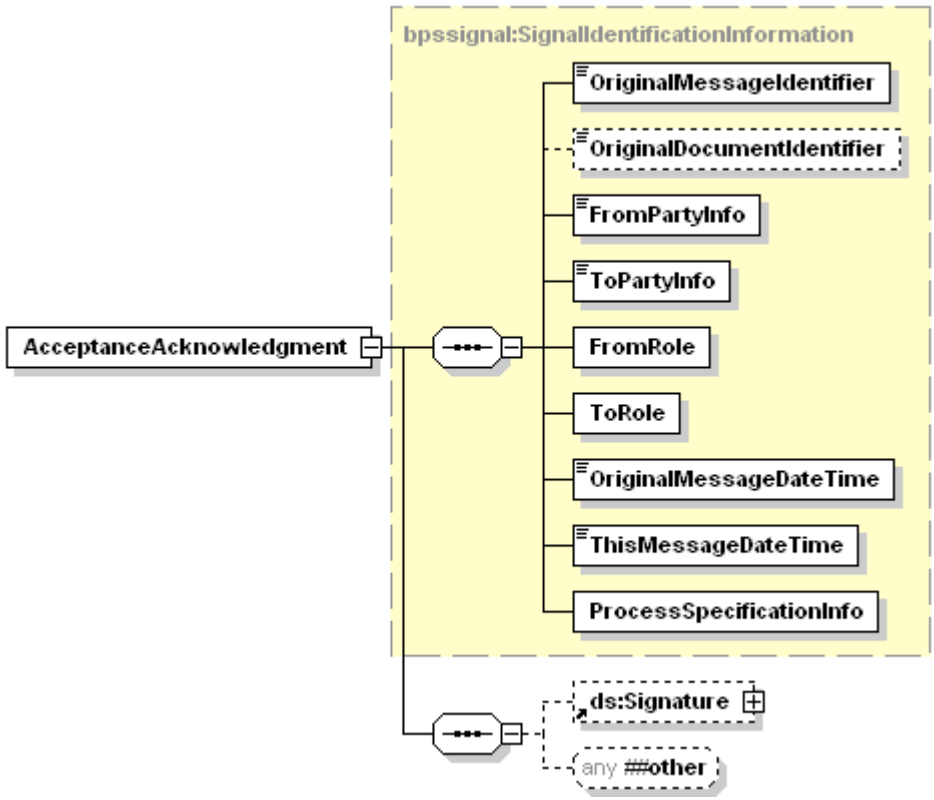
source

```
<xsd:element name="ReceiptAcknowledgment">
  <xsd:complexType>
  <xsd:complexContent>
    <xsd:extension base="bpssignal:SignalIdentificationInformation">
      <xsd:sequence>
        <xsd:element ref="bpssignal:NonRepudiationInformation" minOccurs="0"/>
        <xsd:element ref="ds:Signature" minOccurs="0"/>
        <xsd:any namespace="##other" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
```

2589

7.1.3 AcceptanceAcknowledgement Signal Schema

diagram



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

type extension of **bpssignal:SignalIdentificationInformation**

children **OriginalMessageIdentifier** **OriginalDocumentIdentifier** **FromPartyInfo** **ToPartyInfo** **FromRole** **ToRole** **OriginalMessageDateTime** **ThisMessageDateTime** **ProcessSpecificationInfo** **ds:Signature**

annotation

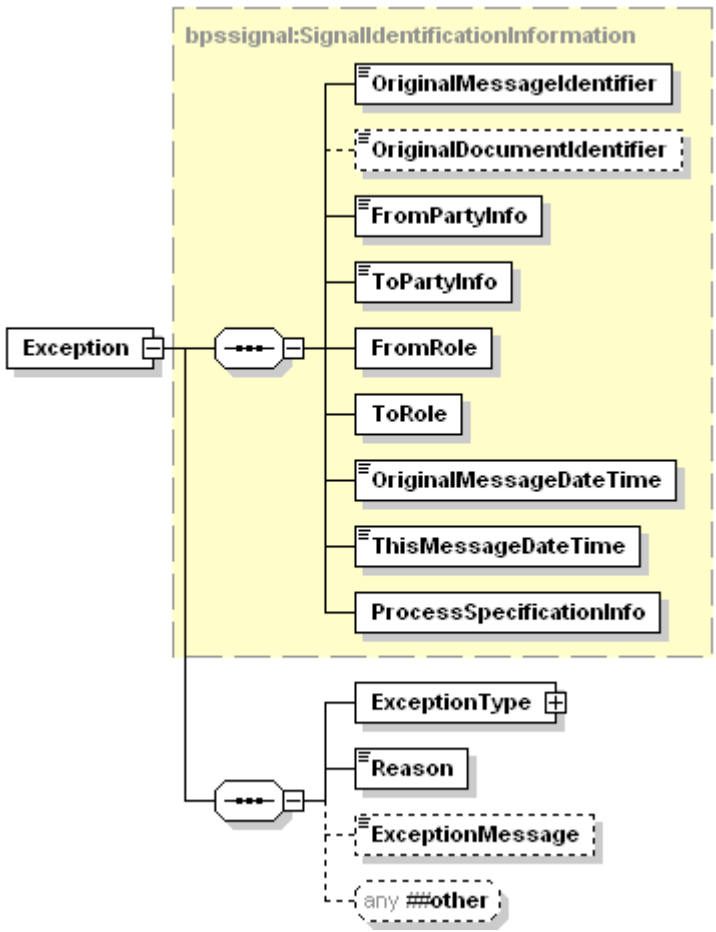
This defines the content structure for messages that need to send an AcceptanceAcknowledgment signals as a business message to a trading partner. Please refer to BPSS document for detailed description of AcceptanceAcknowledgment.
For description of first nine elements, refer to documentation on SignalIdentificationInformation.
If necessary, digital signature can be computed for this signal message and included using "Signature" element from XM Digital Signature namespace.

source

```
<xsd:element name="AcceptanceAcknowledgment">
 <xsd:complexType>
  <xsd:complexContent>
   <xsd:extension base="bpssignal:SignalIdentificationInformation">
    <xsd:sequence>
     <xsd:element ref="ds:Signature" minOccurs="0"/>
     <xsd:any namespace="##other" minOccurs="0"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
```

7.1.4 Exception Signal Schema

diagram



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

type extension of **bpssignal:SignalIdentificationInformation**

children **OriginalMessageIdentifier** **OriginalDocumentIdentifier** **FromPartyInfo** **ToPartyInfo** **FromRole** **ToRole** **OriginalMessageDateTime** **ThisMessageDateTime** **ProcessSpecificationInfo** **ExceptionType** **Reason** **ExceptionMessage**

annotation

This defines the content structure for messages that need to send an exception signals as a business message to a trading partner. For description of first nine elements, refer to documentation on SignalIdentificationInformation.

"ExceptionType" is used to identify various exceptions that can occur during the execution of binary collaboration. Run time processing engine executing BPSS based collaborations generates a "ReceiptException" if request message results in a negative receipt acknowledgment cause of various problems like "Syntax validation" of business message", "Unauthorized execution of process", "Failure of Signature validation in incoming message", "Out of sequence message" corresponding respectively to "Syntax", "Authorization", "Signature", "Sequence".

Run time processing engine executing BPSS based collaborations generates a "AcceptanceException" if request message results in a negative acceptance acknowledgment cause of various problems. Please refer the the specificatior for various reasons why negative acceptance acknowledgment may be sent.

Run time processing engine executing BPSS based collaborations can send a "GeneralException" if processing of a request message results in a state where further processing can not continue.

"Reason" can be used to send a message to convey the reason for exception being generated. "ExceptionMessage" can include a descriptive message corresponding to the exception

source

```
<xsd:element name="Exception">
  <xsd:complexType>
  <xsd:complexContent>
    <xsd:extension base="bpssignal:SignalIdentificationInformation">
      <xsd:sequence>
        <xsd:element name="ExceptionType">
          <xsd:complexType>
            <xsd:choice>
              <xsd:element name="ReceiptException">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
```

```
                    <xsd:enumeration value="Syntax"/>
                    <xsd:enumeration value="Authorization"/>
                    <xsd:enumeration value="Signature"/>
                    <xsd:enumeration value="Sequence"/>
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="AcceptanceException">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Business"/>
                    <xsd:enumeration value="Performance"/>
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="GeneralException">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string"/>
                 </xsd:simpleType>
                </xsd:element>
               </xsd:choice>
              </xsd:complexType>
             </xsd:element>
             <xsd:element name="Reason" type="bpssignal:non-empty-string"/>
             <xsd:element name="ExceptionMessage" type="bpssignal:non-empty-string" minOccurs="0"/>
             <xsd:any namespace="##other" minOccurs="0"/>
            </xsd:sequence>
           </xsd:extension>
          </xsd:complexContent>
         </xsd:complexType>
        </xsd:element>
```

8 EDI support

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

9 Production Rules

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

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

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

1. Classes are rendered as XML elements.
2. Class attributes are rendered as XML attributes. NOTE: occurrence requirements (required vs optional) and default values for attributes are not modeled.

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

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

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

Appendix A: Sample XML Business Process Specification Schema Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessSpecification name="Simple" version="1.1" nameID="Simple-2434134"
xmlns="http://www.ebxml.org/2003/1.1/BusinessProcess" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.ebxml.org/2003/1.1/BusinessProcess
C:\projects\bpss\bpss_1.1\ebBPSS1.08b.xsd">
    <!-- Business Documents -->
    <BusinessDocument name="Catalog Request"
specificationLocation="http://www.xyx.com/CatalogReq.xsd"/>
    <BusinessDocument name="Catalog" specificationLocation="http://www.xyx.com/Catalog.xsd"/>
    <BusinessDocument name="Purchase Order" specificationLocation="http://www.xyx.com/PO.xsd"/>
    <BusinessDocument name="PO Acknowledgement"
specificationLocation="http://www.xyx.com/POAck.xsd"/>
    <BusinessDocument name="Credit Request" specificationLocation="http://www.xyx.com/CreditReq.xsd"/>
    <BusinessDocument name="Credit Confirm" specificationLocation="http://www.xyx.com/CreditCon.xsd"/>
    <BusinessDocument name="ASN" specificationLocation="http://www.xyx.com/CatalogASN.xsd"/>
    <BusinessDocument name="CreditAdvice"
specificationLocation="http://www.xyx.com/CreditAdvice.xsd"/>
    <BusinessDocument name="DebitAdvice" specificationLocation="http://www.xyx.com/DebitAdvice.xsd"/>
    <BusinessDocument name="Invoice" specificationLocation="http://www.xyx.com/Invoice.xsd"/>
    <BusinessDocument name="Payment" specificationLocation="http://www.xyx.com/Payment.xsd"/>
    <BusinessDocument name="Inventory Report Request"
specificationLocation="http://www.xyx.com/InvReq.xsd"/>
    <BusinessDocument name="Inventory Report" specificationLocation="http://www.xyx.com/InvRep.xsd"/>
    <Package name="Ordering">
        <!-- Here are all the Business Transactions needed -->
        <BusinessTransaction name="Catalog Request">
            <RequestingBusinessActivity name="RequestCatalog">
                <DocumentEnvelope businessDocument="Catalog Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="SendCatalog">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="Catalog"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Create Order">
            <RequestingBusinessActivity name="SendOrder" isNonRepudiationRequired="true"
timeToAcknowledgeReceipt="P2D" timeToAcknowledgeAcceptance="P3D">
                <DocumentEnvelope businessDocument="Purchase Order"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="SendPOAcknowledgement"
isNonRepudiationRequired="true" timeToAcknowledgeReceipt="P5D">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="PO
Acknowledgement"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Check Credit ">
            <RequestingBusinessActivity name="CreditCheck">
                <DocumentEnvelope businessDocument="Credit Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="ConfirmCredit">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="Credit Confirm"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Notify of advance shipment">
            <RequestingBusinessActivity name="AdvanceShipmentNotification">
                <DocumentEnvelope businessDocument="ASN"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="ASNResponse"/>
        </BusinessTransaction>
        <BusinessTransaction name="Process Credit Payment">
            <RequestingBusinessActivity name="CreditPaymentProcess">
                <DocumentEnvelope businessDocument="CreditAdvice"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="CreditPaymentProcessResponse">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="DebitAdvice"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
```

```
        <BusinessTransaction name="Process Payment">
            <RequestingBusinessActivity name="PaymentProcess">
                <DocumentEnvelope businessDocument="Invoice"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="SendPayment">
                <DocumentEnvelope isPositiveResponse="true" businessDocument="Payment"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Request Inventory Report">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope businessDocument="Inventory Report Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="Inventory Report">
                <DocumentEnvelope businessDocument="Inventory Report"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <!-- Now the Binary Collaborations -->
        <BinaryCollaboration name="Request Catalog" initiatingRoleID="1122B1">
            <Role name="requestor" nameID="1122B1"/>
            <Role name="provider" nameID="2211A1"/>
            <Start toBusinessState="Catalog Request"/>
            <BusinessTransactionActivity name="Catalog Request" businessTransaction="Catalog Request"
fromRole="requestor" toRole="provider"/>
            <Success fromBusinessState="Catalog Request" conditionGuard="Success"/>
            <Failure fromBusinessState="Catalog Request" conditionGuard="Failure"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Firm Order" timeToPerform="P2D" initiatingRoleID="1122B2">
            <Documentation>timeToPerform = Period: 2 days from start of transaction</Documentation>
            <Role name="buyer" nameID="1122B2"/>
            <Role name="seller" nameID="1122B3"/>
            <Start toBusinessState="Create Order"/>
            <BusinessTransactionActivity name="Create Order" businessTransaction="Create Order"
fromRole="buyer" toRole="seller"/>
            <Success fromBusinessState="Create Order" conditionGuard="Success"/>
            <Failure fromBusinessState="Create Order" conditionGuard="Failure"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Product Fulfillment" timeToPerform="P5D" initiatingRoleID="1122B2">
            <Documentation>timeToPerform = Period: 5 days from start of transaction</Documentation>
            <Role name="buyer" nameID="1122B2"/>
            <Role name="seller" nameID="1122B3"/>
            <Start toBusinessState="Create Order"/>
            <BusinessTransactionActivity name="Create Order" businessTransaction="Create Order"
fromRole="buyer" toRole="seller"/>
            <BusinessTransactionActivity name="Notify shipment" businessTransaction="Notify of advance
shipment" fromRole="seller" toRole="buyer"/>
            <Success fromBusinessState="Notify shipment" conditionGuard="Success"/>
            <Failure fromBusinessState="Notify shipment" conditionGuard="Failure"/>
            <Transition fromBusinessState="Create Order" toBusinessState="Notify shipment"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Inventory Status" initiatingRoleID="1122B1">
            <Role name="requestor" nameID="1122B1"/>
            <Role name="provider" nameID="2211A1"/>
            <Start toBusinessState="Inventory Report Request"/>
            <BusinessTransactionActivity name="Inventory Report Request" businessTransaction="Inventory
Report Request" fromRole="requestor" toRole="provider"/>
            <Success fromBusinessState=" Inventory Report Request " conditionGuard="Success"/>
            <Failure fromBusinessState=" Inventory Report Request " conditionGuard="Failure"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Credit Inquiry" initiatingRoleID="9122B1">
            <Role name="creditor" nameID="9122B1"/>
            <Role name="credit service" nameID="8122B1"/>
            <Start toBusinessState="Check Credit"/>
            <BusinessTransactionActivity name="Check Credit" businessTransaction="Check Credit"
fromRole="creditor" toRole="credit service"/>
            <Success fromBusinessState="Check Credit" conditionGuard="Success"/>
            <Failure fromBusinessState="Check Credit" conditionGuard="Failure"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Credit Payment" initiatingRoleID="6122B1">
            <Role name="payee" nameID="6122B1"/>
            <Role name="payor" nameID="7122B1"/>
            <Start toBusinessState="Process Credit Payment"/>
```

```
            <BusinessTransactionActivity name="Process Credit Payment" businessTransaction="Process
Credit Payment" fromRole="payee" toRole="payor"/>
            <Success fromBusinessState="Process Credit Payment" conditionGuard="Success"/>
            <Failure fromBusinessState="Process Credit Payment" conditionGuard="Failure"/>
        </BinaryCollaboration>
        <!-- A compound BinaryCollaboration for illustration purposes-->
        <BinaryCollaboration name="Credit Charge" initiatingRoleID="8132B1">
            <Role name="charger" nameID="8132B1"/>
            <Role name="credit service" nameID="8122B1"/>
            <Start toBusinessState="Credit Inquiry"/>
            <CollaborationActivity name="Credit Inquiry" binaryCollaboration="Credit Inquiry"
fromRole="charger" toRole="credit service"/>
            <CollaborationActivity name="Credit Payment" binaryCollaboration="Credit Payment"
fromRole="charger" toRole="payor"/>
            <Success fromBusinessState="Credit Payment" conditionGuard="Success"/>
            <Failure fromBusinessState="Credit Payment" conditionGuard="Failure"/>
            <Transition fromBusinessState="Credit Inquiry" toBusinessState="Credit Payment"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Fulfillment Payment" initiatingRoleID="6122B1">
            <Role name="payee" nameID="6122B1"/>
            <Role name="payor" nameID="7122B1"/>
            <Start toBusinessState="Process Payment"/>
            <BusinessTransactionActivity name="Process Payment" businessTransaction="Process
Payment" fromRole="payee" toRole="payor"/>
            <Success fromBusinessState="Process Payment" conditionGuard="Success"/>
            <Failure fromBusinessState="Process Payment" conditionGuard="Failure"/>
        </BinaryCollaboration>
        <!-- First the overall MultiParty Collaboration -->
        <MultiPartyCollaboration name="DropShip">
            <BusinessPartnerRole name="Customer">
                <Performs role="requestor" roleIDREF="1122B1"/>
                <Performs role="buyer" roleIDREF="1122B2"/>
                <Transition fromBusinessState="Catalog Request" toBusinessState="Create Order"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="Retailer">
                <Performs role="provider" roleIDREF="2211A1"/>
                <Performs role="seller" roleIDREF="1122B3"/>
                <Performs role="creditor" roleIDREF="9122B1"/>
                <Performs role="buyer" roleIDREF="1122B2"/>
                <Performs role="payee" roleIDREF="6122B1"/>
                <Performs role="payor" roleIDREF="7122B1"/>
                <Performs role="requestor" roleIDREF="1122B1"/>
                <Transition fromBusinessState="Create Order" toBusinessState="Check Credit"/>
                <Transition fromBusinessState="Check Credit" toBusinessState="Credit Payment"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="DropShip Vendor">
                <Performs role="seller" roleIDREF="1122B3"/>
                <Performs role="payee" roleIDREF="6122B1"/>
                <Performs role="provider" roleIDREF="2211A1"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="Credit Authority">
                <Performs role="credit service" roleIDREF="8122B1"/>
                <Performs role="payor" roleIDREF="7122B1"/>
            </BusinessPartnerRole>
        </MultiPartyCollaboration>
    </Package>
</ProcessSpecification>
```

Appendix B: Sample XML Signals

```
<?xml version="1.0" encoding="UTF-8"?>
<bpssignal:ReceiptAcknowledgment
xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsd:schemaLocation="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS  BPSS_Signals.xsd">
```

```
        <bpssignal:OriginalMessageIdentifier>MessageIdentifier-1</bpssignal:OriginalMessageIdentifier>
        <bpssignal:FromPartyInfo bpssignal:type="DUNS.com">PartyA</bpssignal:FromPartyInfo>
        <bpssignal:ToPartyInfo bpssignal:type="DUNS.com">PartyB</bpssignal:ToPartyInfo>
        <bpssignal:FromRole bpssignal:name="Buyer" xlink:type="simple"
xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
        <bpssignal:ToRole bpssignal:name="Seller" xlink:type="simple"
xlink:href="http://www.rosettanet.org/processes/3A4.xml#Seller"/>
        <bpssignal:OriginalMessageDateTime>2002-03-05T19:00:00</bpssignal:OriginalMessageDateTime>
        <bpssignal:ThisMessageDateTime>2002-03-05T20:00:00</bpssignal:ThisMessageDateTime>
        <bpssignal:ProcessSpecificationInfo bpssignal:version="2.0"
bpssignal:name="PIP3A4RequestPurchaseOrder" xlink:type="simple"
xlink:href="http://www.rosettanet.org/processes/3A4.xml"
bpssignal:nameID="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
        <bpssignal:NonRepudiationInformation>
            <bpssignal:MessagePartNRInformation>
                <ds:Reference ds:URI="cid://Message-Part-1">
                    <ds:DigestMethod ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    <ds:DigestValue>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8bd012</ds:DigestValue>
                </ds:Reference>
            </bpssignal:MessagePartNRInformation>
            <bpssignal:MessagePartNRInformation>
                <ds:Reference ds:URI="cid://Message-Part-2">
                    <ds:DigestMethod ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    <ds:DigestValue>afde1AbcgGSALMAAAQCAEMmCZtuMFQxDS8be</ds:DigestValue>
                </ds:Reference>
            </bpssignal:MessagePartNRInformation>
            <bpssignal:MessagePartNRInformation>
                <bpssignal:MessagePartIdentifier>Message-Part-3</bpssignal:MessagePartIdentifier>
            </bpssignal:MessagePartNRInformation>
        </bpssignal:NonRepudiationInformation>
    </bpssignal:ReceiptAcknowledgment>


    <?xml version="1.0" encoding="UTF-8"?>
    <bpssignal:AcceptanceAcknowledgment
    xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
    xsd:schemaLocation="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS  BPSS_Signals.xsd">
        <bpssignal:OriginalMessageIdentifier>MessageIdentifier-1</bpssignal:OriginalMessageIdentifier>
        <bpssignal:FromPartyInfo bpssignal:type="DUNS.com">PartyA</bpssignal:FromPartyInfo>
        <bpssignal:ToPartyInfo bpssignal:type="DUNS.com">PartyB</bpssignal:ToPartyInfo>
        <bpssignal:FromRole bpssignal:name="Buyer" xlink:type="simple"
    xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
        <bpssignal:ToRole bpssignal:name="Seller" xlink:type="simple"
    xlink:href="http://www.rosettanet.org/processes/3A4.xml#Seller"/>
        <bpssignal:OriginalMessageDateTime>2002-03-05T19:00:00</bpssignal:OriginalMessageDateTime>
        <bpssignal:ThisMessageDateTime>2002-03-05T20:00:00</bpssignal:ThisMessageDateTime>
        <bpssignal:ProcessSpecificationInfo bpssignal:version="2.0"
    bpssignal:name="PIP3A4RequestPurchaseOrder" xlink:type="simple"
    xlink:href="http://www.rosettanet.org/processes/3A4.xml"
    bpssignal:nameID="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
    </bpssignal:AcceptanceAcknowledgment>

    <?xml version="1.0" encoding="UTF-8"?>
    <bpssignal:Exception xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
    xsd:schemaLocation="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS
    BPSS_Signals.xsd">
        <bpssignal:OriginalMessageIdentifier>MessageIdentifier-1</bpssignal:OriginalMessageIdentifier>
        <bpssignal:FromPartyInfo bpssignal:type="DUNS.com">PartyA</bpssignal:FromPartyInfo>
        <bpssignal:ToPartyInfo bpssignal:type="DUNS.com">PartyB</bpssignal:ToPartyInfo>
        <bpssignal:FromRole bpssignal:name="Buyer" xlink:type="simple"
    xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
        <bpssignal:ToRole bpssignal:name="Seller" xlink:type="simple"
    xlink:href="http://www.rosettanet.org/processes/3A4.xml#Seller"/>
        <bpssignal:OriginalMessageDateTime>2002-03-05T19:00:00</bpssignal:OriginalMessageDateTime>
        <bpssignal:ThisMessageDateTime>2002-03-05T20:00:00</bpssignal:ThisMessageDateTime>
        <bpssignal:ProcessSpecificationInfo bpssignal:version="2.0"
    bpssignal:name="PIP3A4RequestPurchaseOrder" xlink:type="simple"
```

```
xlink:href="http://www.rosettanet.org/processes/3A4.xml"
bpssignal:nameID="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
    <bpssignal:ExceptionType>
        <bpssignal:ReceiptException>Signature</bpssignal:ReceiptException>
    </bpssignal:ExceptionType>
    <bpssignal:Reason>State transition failure</bpssignal:Reason>
    <bpssignal:ExceptionMessage>Signature Validation Failed for request
message</bpssignal:ExceptionMessage>
</bpssignal:Exception>
```

10 References

1. UN/CEFACT Modeling Methodology - Meta Model - Revision 12 (2003-01-17) specification, http://webster.disa.org/cefact-groups/tmg/downloads/general/approved/UMM-MM-V20030117.zip
2. UN/CEFACT Core Components Technical Specification, Version 2.0, http://webster.disa.org/cefact-groups/tmg/downloads/general/approved/CEFACT-CCTS-Version-2pt0.zip
3. ebXML Technical Architecture Specification, version 1.04, http://www.ebxml.org/specs/ebTA.pdf
4. Key Words for use in RFCs to Indicate Requirement Levels, Internet Engineering Task Force RFC 2119, http://www.ietf.org/rfc/rfc2119.txt.
5. Extensible Markup Language (XML), World Wide Web Consortium, http://www.w3.org/XML.
6. XML Schema Part 1: Structures, Worldwide Web Consortium, http://www.w3.org/TR/xmlschema-1/.
7. XML Schema Part 2: Datatypes, Worldwide Web Consortium,http://www.w3.org/TR/xmlschema-2/.
8. ebXML Message Service Specification, http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.
9. ebXML Registry Services Specification, http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf.
10. ebXML Collaboration-Protocol Profile and Agreement Specification V1.9, http://www.oasis-open.org/committees/ebxml-cppa/documents/working_drafts/ebCPP-1_9.pdf
11. Multipurpose Internet Mail Extensions (MIME) Part One, IETF RFC 2045: Format of Internet Message Bodies, N. Freed, N. Borenstein, Authors. Internet Engineering Task Force, November 1996. Available at http://www.ietf.org/rfc/rfc2045.txt

11 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.

12 Contact Information

TMG Chair:
Klaus-Dieter Naujok
Global e-Business Advisory Council
e-mail: klausn@attglobal.net

13 Copyright Statement