ebXML Business Process Specification Schema Technical Specification v2.0.4

OASIS Standard, 21 December 2006

Specification URIs:

This Version:

docs.oasis-open.org/ebxml-bp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en.doc
docs.oasis-open.org/ebxml-bp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en-html/
docs.oasis-open.org/ebxml-bp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en.odt
docs.oasis-open.org/ebxml-bp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en.pdf

Previous Version:

docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-cs-en.odt
docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-cs-en.pdf

Latest Version:

docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-os-en.doc
docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-os-en.html
docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-os-en.odt
docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-os-en.pdf

Technical Committee:

ebXML Business Process Technical Committee

Co-chairs:

Dale Moberg, Cyclone Commerce/Axway
Monica J. Martin, Sun Microsystems

Editors:

Jean-Jacques Dubray, Individual, jdubray@gmail.com [previous member]
Sally St. Amand, Individual, sallystamand@yahoo.com
Monica J. Martin, Sun Microsystems, monica.martin@sun.com

Contributors:

John Yunker, Individual yunker@amazon.com (previous member)
David Webber, Individual, <david@drrw.info>
Dale Moberg, Cyclone Commerce/Axway, co-chair, dmoberg@us.axway.com
Kenji Nagahashi, Fujitsu, nagahashi@us.fujitsu.com
Stephen Green, Individual, stephengreenubl@gmail.com (previous member)
Sacha Schlegel, Individual, sacha@schlegel.li
Monica J. Martin, Sun Microsystems, co-chair, monica.martin@sun.com

Contributions for the development of ebBP examples of UBL related documents by J. Dean Hemopo, ebxml-dev, New Zealand (user community), and Stephen Green, UK local government (user community) and Sacha Schlegel (Member).

Related Work:

See Section 1.4 : Related Documents.

Abstract:

This document defines a standards-based business process foundation that promotes the automation and predictable exchange of Business Collaboration definitions using XML.

Status:

This set of ebBP documents are compatible with the ebXML Business Process Specification Schema v1.01 technical specification and schema, and a migration path is possible from v1.01, v1.04 and v1.05 to v2.0.x documents. The technical specification supersedes the v2.0 Committee Draft / Committee Specification[1], v2.0.1 and v2.0.2 Committee Drafts, and the v2.0.3 Committee Specification.

Six packages are provided for ebBP:

1. Normative: A package for the technical specification and appendices (Artifact Type: Spec, and Artifact Type: Spec and Descriptive Name: Appendices)
2. Normative: A package for the core schema (Artifact Type: Schema)
3. Normative: A package for the Business Signal schema (Artifact Type: Schema, Descriptive Name: SignalSchema)
4. Non-normative: A package that includes the Public Review comments list, files for an exemplary XSLT transform to assist the user community to begin to migrate v1.01, v1.04 and v1.05 ebBP instances (for information and reference only) [Artifact Type: Document, Descriptive Name: Supplements]
5. Normative: A package of ebBP schema-generated documentation for ebBP schema (Artifact Type: Document, Descriptive Name: Schema)
6. Normative: A package of ebBP signal schema-generated documentation (Artifact Type: Document, Descriptive Name: SignalSchema).

These documents are updated periodically. Send comments to the editor.

Note: The schemas (core and signals) are also located individually outside of the packages as specified in Section 2.

Exemplary process definition and signal instances and illustrations are also provided in a publicly available package on the OASIS site. This final package is non-normative and outside the review and TC process cycle of this technical specification. This technical specification provides non-normative examples (XML instance snippets) while more complex ebBP definitions may be found in the examples package.

The ebXML Business Process TC charter including scope is found at: http://www.oasis-open.org/committees/ebxml-bp/charter.php.

Committee members should send comments on this specification to the ebxml-bp@lists.oasis-open.org list. Others should subscribe to and send comments to the ebxml-bp-comment@lists.oasis-open.org list. To subscribe, send an email message to ebxml-bp-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the ebXML Business Process TC web page (http://www.oasis-open.org/committees/ebxml-bp/ipr.php). The IPR policy in effect as of this document is the Legacy IPR policy.

The non-normative errata page for this specification is located at www.oasis-open.org/committees/ebxml-bp.

[1] The preceding OASIS TC process indicates Committee Specification while the new TC process indicates Committee Draft followed by a Committee Specification. The v2.0 packages were applicable under the old TC process as the quorate TC vote was initiated prior to the effective date of the new TC process (although the vote concluded after 15 April 2005). Under the new TC process, this document is a Committee Draft.

Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

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

Table of Contents

Note: Appendices are held in a separate document in the Spec package.

1 Introduction

The eBusiness eXtensible Markup Language (ebXML) Business Process Specification Schema (BPSS) 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 10. In the future, when a reference guide becomes available subsequent versions will be evaluated and other metadata requirements analyzed. These could include those developed under the United Nations Centre for Trade and Facilitation and Electronic Business (UN/CEFACT), such as from the Unified Business Agreements and Contracts (UBAC).[2] The ebBP technical specification supports the specification of Business Transactions and the choreography of Business Transactions into Business Collaborations. All Business Transactions are 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 ebBP technical specification addresses Business Collaborations between any number of parties (Business Collaborations specialized to Binary or Multiparty Collaborations). It also enables participants, which are capable of using Web service or combined technologies (such as ebXML and web services) to participate in a Business Collaboration. It is anticipated that a subsequent version of this technical specification will address additional features such as the semantics of economic exchanges and contracts, and context based content based on the metadata requirements provided by relevant organizations.

> *Implementation Note:*
>
> *Throughout this document, shorthand is used. The technical specification is referenced as the ebBP technical specification. An ebBP business process definition is identified as an ebBP definition. An ebXML BPSS instance is an ebBP instance. An ebXML BPSS schema is an ebBP schema.*

1.1 Terminology

The key WORDS *MUST*, *MUST NOT*, *REQUIRED*, *SHALL*, *SHALL NOT*, *SHOULD*, *SHOULD NOT*, *RECOMMENDED*, *MAY*, and *OPTIONAL* in this document are to be interpreted as described in **[RFC 2119]**. These provide indications as to normative capabilities defined in this technical specification.

1.2 Summary of Contents of Document

This document describes the ebBP technical specification.

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

1.3 Audience

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

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

[2] A reference will be available when these documents are published or publicly available.

1.4 Related Documents

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

- ebXML Technical Architecture Specification, version 1.04
- ebXML Core Components Technical Specification, version 2.01
- ebXML Collaboration-Protocol Profile and Agreement Specification version 2.1 errata
- 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 (original)
- UN/CEFACT - Common Business Process Catalog Technical Specification, version 1.0 (updated)
- ebXML Message Service Specification version 2.0
- UN/CEFACT Modeling Methodology (UMM) as defined in the N090R10 metamodel and reference specification

1.5 Normative References

[XML] Extensible Markup Language (XML), World Wide Web Consortium, http://www.w3.org/XML.

[XSD1] XML Schema Part 1: Structures, Worldwide Web Consortium, http://www.w3.org/TR/xmlschema-1/.

[XSD2] XML Schema Part 2: Datatypes, Worldwide Web Consortium, http://www.w3.org/TR/xmlschema-2/.

[XInclude] XInclude, Recommendation, W3C, 20 December 2004**:** http://www.w3.org/TR/xinclude.

[RFC2119] S. Bradner. Request for Comments 2119, Key words for use in RFCs to Indicate Requirement Levels. IETF (Internet Engineering Task Force). 1997. Internet Engineering Task Force RFC 2119, http://www.ietf.org/rfc/rfc2119.txt.

[XPath] XML Path Language (XPath), W3C Recommendation, 16 November 1999, http://www.w3.org/TR/xpath.

[RFC2396] T. Berners-Lee. Request for Comments 2396, Uniform Resource Identifiers (URI): Generic Syntax. IETF (Internet Engineering Task Force). 1998. Internet Engineering Task Force RFC 2396, http://www.ietf.org/rfc/rfc2396.txt.

1.6 Non-Normative References

[BPAW] ebXML Business Process Analysis Worksheets & Guidelines, v1.0, http://www.ebxml.org/specs/bpWS.pdf.

[BPBIA] ebXML Business Process and Business Information Analysis Overview, v1.0, http://www.ebxml.org/specs/bpOVER.pdf.

[BPMN] Business Process Modeling Notation (BPMN) v1.0, Object Management Group (OMG), at: www.bpmn.org (BPMN site) or http://www.omg.org/docs/dtc/06-02-01.pdf (at OMG).

[CBPC1] (original) ebXML Catalog of Common Business Processes, v1.0, http://www.ebxml.org/specs/bpPROC.pdf.

[CBPC2] (updated) UN/CEFACT - Common Business Process Catalog Technical Specification, v1.0, 30 September 2005, http://www.cen.eu/UNcEFACTforum/TBG/tbg14.htm.

[DocEng] Glushko, Robert and Tim McGrath. Document Engineering - Analyzing and Designing Documents for Business Informatics and Web Services, http://www.docengineering.com/.

[ebCCTS] ISO/TS 15000-5:2005 Electronic Business Extensible Markup Language (ebXML) — Part 5: ebXML Core Components Technical Specification, v 2.01 (ebCCTS), http://www.oasis-open.org/committees/download.php/6232/CEFACT-CCTS-Version-2pt01.zip.

[ebCPPA2.1] ebXML Collaboration-Protocol Profile and Agreement working editor's draft errata, v2.1, 13 July 2005, http://lists.oasis-open.org/archives/ebxml-cppa/200507/msg00000.html. Note: The .zip file is included in message. At the time of this technical specification the schema is under revision related to CPA changes.

[ebCPPA2] ebXML Collaboration-Protocol Profile and Agreement Specification v2.0, 20 May 2002, http://www.oasis-open.org/committees/download.php/202/ebCPP-2_0.pdf.

[ebMS2] ebXML Message Service Specification, v2.0, http://www.oasis-open.org/committees/document.php?document_id=5553&wg_abbrev=ebxml-msg.

[ebRIM3] ebXML Registry Information Model OASIS Standard, v3.0, 5 May 2005, http://docs.oasis-open.org/regrep/v3.0/regrep-3.0-os.zip.

[ebRS3] ebXML Registry Services OASIS Standard, v3.0, 5 May 2005, http://docs.oasis-open.org/regrep/v3.0/regrep-3.0-os.zip.

[ebTA] ebXML Technical Architecture Specification, v1.04, http://www.ebxml.org/specs/ebTA.pdf.

[ecPAT] ebXML E-Commerce Patterns, v1.0, http://www.ebxml.org/specs/bpPATT.pdf.

[MIME] 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

[RNIF] RosettaNet Implementation Framework: Core Specification, Vv1.0: Release 2.00.00, 13 July 2001.

[SCH] Schematron, published ISO standard (DSDL project, www. dsdl.org), ISO/IEC 19757 - DSDL Document Schema Definition Language - Part 3: Rule-based validation - Schematron, http://xml.ascc.net/resource/schematron/schematron.html, http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40833.

[UMM] UN/CEFACT Modelling Methodology - Meta Model and Reference Information - Revision 10, N090 (2001-11-01) specification, http://www.untmg.org/index.php?option=com_docman&task=docclick&Itemid=137&bid=21&limitstart=0&limit=5 (as of September 2006).

[WS-A] WS-Addressing, W3C, W3C Recommendation, May 2006, http://www.w3.org/2005/08/addressing.

[WSDL1.1] Web Services Description Language, v1.1, W3C Note, 15 March 2001, http://www.w3.org/TR/wsdl.

[WSDL2]	Web Services Description Language, v2.0, Candidate Recommendation, 27 March 2006, http://www.w3.org/TR/wsdl20/.
[XSLT]	XML Transformations (XSLT), W3C Recommendation, v1.0, 16 November 1999, http://www.w3.org/TR/xslt.

2 Design Objectives

2.1 Goals/Objectives/Requirements/Problem Description

ebBP definitions describe interoperable business processes that allow business partners to collaborate and achieve a given business goal. These definitions MUST be executed by software components that collaborate on behalf of the business partners.

The goal of the ebBP technical specification is to provide the bridge between eBusiness process modeling and execution of eBusiness software components.

The ebBP technical specification provides for the nominal set of specification elements necessary to specify a Business Collaboration between business partners, and to provide configuration parameters for the partners' runtime systems in order to execute that Business Collaboration between a set of eBusiness software components.

A business process definition created with the ebBP technical specification is referred to as an ebBP definition.

The ebBP technical specification is available as an XML Schema (http://www.w3.org/2001/XMLSchema). The ebBP XML schema, that provides the specification for XML based ebBP definitions, can be found at this location:

http://docs.oasis-open.org/ebxml-bp/ebbp-2.0

(schema: ebbp-2.0.4.xsd)

The ebBP XML signal schema can be found at this location:

http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0

(signal schema: ebbp-signals-2.0.4.xsd)

In order to accommodate varying tool capabilities surrounding namespaces and directories using URIs, the URI for each schema has been updated. Current URI paths are found on the OASIS ebBP public web site at:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebxml-bp

Under "'Technical Work Produced by the Committee'"

The schemas reflect the latest computable formats for an ebBP process definition.

2.2 Caveats and Assumptions

This technical specification is designed to specify the run time aspects of a Business Collaboration.

It is not intended to incorporate a methodology, and does not directly prescribe the use of a methodology. This specification does not by itself define Business Documents Structures. It is intended to work in conjunction with already existing Business Document definitions, and/or the document metamodel defined by the ebXML Core Components specifications.

The ebBP technical specification recognizes and concretely expresses the six defined, Business Transaction patterns-Commercial Transaction, Notification, Information Distribution, Request-Response, Request-Confirm, and Query Response. In the future, it is expected that new or additional business requirements (such as for metadata) may be defined for contractual agreements, acceptance, revocation of offers, etc. through efforts such as that of UN/CEFACT at a minimum.

Examples, sample instances and the glossary are non-normative in this technical specification. They are provided to aid the user community and implementers to use the ebBP v2.0.4 technical specification and associated schemas. In addition to portions of this technical specification, the ebBP and Business Signal schemas are related to and normative to this technical specification. The examples are held outside of the non-normative and normative packages to enable frequent updates.

2.3 Detailed Specification of Model Components

As with all the other specifications in the ebXML framework, an ebBP process definition may be effectively used with other technologies. The ebXML framework has been composed of several independent, but related or aligned, components. Specifications for each component can be used individually, composed as desired, or integrated with other evolving technologies.

From the onset, these specifications have sought to be aligned as much as practical and capable of being composed together and used with other technologies. That flexibility and composability are important aspects not only to the adoption of these standards but their effective use and successful deployment into heterogeneous environments and across domains. In the context of this technical specification, Business Collaborations may be executed using the ebBP process definition and/or used with other technologies. As it relates to the other specifications in the ebXML framework, an ebBP process definition supports the loose coupling and alignment needed to execute Business Collaborations. This specification may also be used when several other software components are used to enable the execution of Business Collaborations. One example is the use of web services mapped to business transactions activities. The ebBP technical specification is used to specify the business process related configuration parameters for configuring a Business Service Interface (BSI) to execute and monitor these collaborations. The ebBP business semantics and syntax are also well-suited to enable definition of modular process building blocks that are combined in complex activities to meet user community needs.

This section discusses:

- How the ebBP technical specification fits in with other ebXML specifications and may be used with other emerging technologies (such as WSDL). An ebBP process definition does not preclude composition with other process related technologies.
- How to use the ebBP artifacts at design time, either for specifying brand new collaborations and transactions, or for re-using existing ones.
- How to specify core transaction semantics and parameters needed for or that may be used by a Collaboration-Protocol Profile (CPP) or Collaboration Protocol Agreement (CPA).
- Run-time transaction and collaboration semantics that the ebBP schema specifies and the BSI is expected to manage.

As this technology matures and relevant profiles emerge, more compatibility points will be specified or conformance information (where appropriate and applicable) defined in the context of heterogeneous technology integration. For example, an ebBP profile is under development in OASIS ebXML Implementation, Interoperability Conformance (IIC) TC, based on their deployment template.

2.3.1 Use of ebBP With Other Specifications

The ebBP technical specification provides the structure and semantics of Business Collaboration definitions.

A Business Collaboration consists of a set of roles that collaborate by exchanging Business Documents through a set of choreographed transactions.

As shown in the following figure, Business Documents are defined at the intersection between the ebBP technical specification and the ebXML Core Component specifications. An ebBP definition will reference, but not define, a set of logical Business Documents. Within an ebBP definition, Business Documents are either defined by some external document specification, or assembled from lower level information structures called core components. The assembly is based on a set of contexts, many of which are provided by the business processes, i.e. collaborations that use the documents in their Document Flows.

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

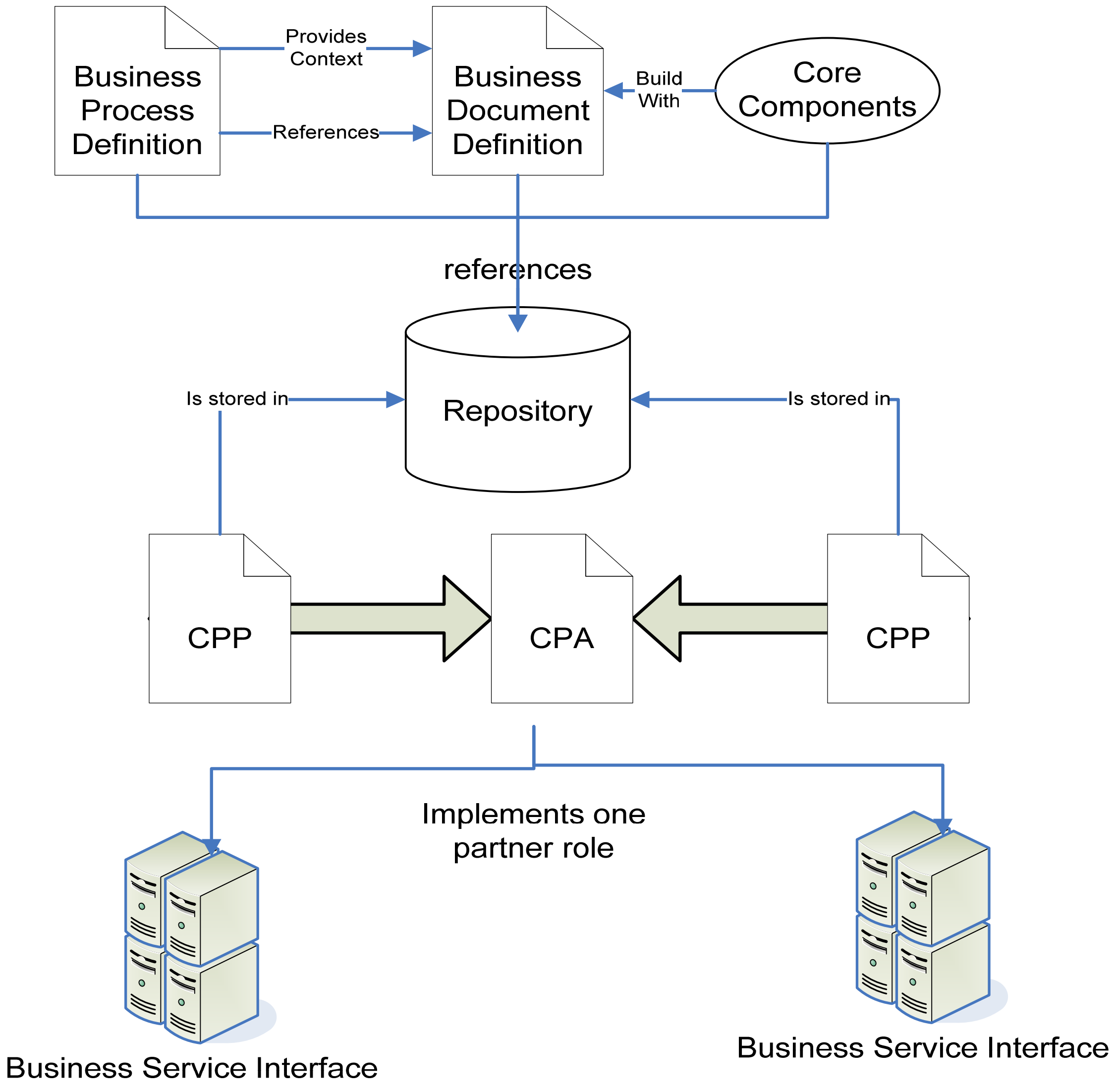

Business Process Definition
Provides Context
References
Business Document Definition
Build With
Core Components
references
Is stored in
Repository
Is stored in
CPP
CPA
CPP
Implements one partner role
Business Service Interface
Business Service Interface

Figure 1: ebBP Definition and other ebXML artifacts

The user will extract and transform the necessary information from an existing Business Process and Information Model and create an XML representation of an ebBP definition.

The XML representation of the ebBP definition gets stored in the ebXML repository and registered in the ebXML registry for future retrieval. The ebBP definition would be registered using classifiers derived during its design.

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

If one or more parties wish to participate on the basis of one or more web service definitions the corresponding WSDL file(s) associated to the BTA(s) that is(are) representing the party MAY be generated and MAY be referenced in the CPA if necessary.

Guided by the CPP and CPA specifications the resulting XML document then MAY become the configuration file for one or more BSI, i.e. the software component that MAY manage either business partner's participation in a Business Collaboration.

2.4 Relationship to Other Specifications and Standards

This section describes the relationship of ebBP technical specification to other specifications and/or standards. Later in Section 3, use of this specification with CPA is discussed in further detail.

2.4.1 Relationship to CPP/CPA

An ebBP definition is, along with protocol specifications, the object of the agreement between two or more parties. The ebBP definition MAY therefore be incorporated with or referenced by ebXML trading partner CPP or CPA. The CPA articulates the technical mechanisms that configure a runtime system and encourage interoperability between two parties that may use different applications or software from different vendors.

Each CPP MAY declare its support for one or more Roles within the ebBP definition. An ebBP definition is also a machine interpretable specification needed for a BSI, 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. The CPA currently supports the notion of business transactions between collaborating roles.

Messaging and CPA support conversations between parties. Each individually and collectively map to the ebBP. The ebBP schema (and technical specification) provides guidance to the CPA and messaging service regarding the processes used, the constraints expected, and the relationship that exists between the parties.

2.4.2 Relationship to Core Components

The ebBP technical specification does not by itself support the definition of Business Documents. Rather, an ebBP definition merely points to the definition of logical Business Documents.[3] Such definitions MAY either be XML based, or – as attachments – MAY be any other structure, or completely unstructured (e.g. related to images, EDI, binary data). XML based Business Document Specifications MAY be based on the ebXML Core Components Technical Specification (CCTS) such as OASIS Universal Business Language (UBL) specifications. In the addition to the non-normative appendices to this technical specification, example instance will be included in a separate package, publicly available on the OASIS web site to aid user communities. These examples or illustrations of ebBP v2.0.4 instance use relevant document

[3] Specification elements related to a logical Business Document if further defined in Section 3.4.6.2.

vocabularies such as UBL and its corresponding Small Business Subset (SBS) to equate the use of ebBP in real-world scenarios.

In ebBP, transitions are handled by state changes, whether sequential or determined through the transitions. These transition conditions MAY relate to the sequential ordering handled by the messaging and where those ebBP expectations MAY be enforced. The relationship between the Messaging Service Interface and the BSI are further described in the appendices to this technical specification.

2.4.3 Relationship to ebXML Message Service Specification

The ebBP technical specification 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.

Messaging and CPA support conversations between parties. Each individually and collectively may map to the ebBP. The ebBP schema (and technical specification) provides guidance to the CPA and messaging service regarding the processes used, the constraints expected, and the relationship that exists between the parties.

2.4.4 Relationship to WSDL

This version of the ebBP technical specification provides a mapping between BTAs (i.e. the usage of a Business Transaction definition in a Business Collaboration definition) and operations of one or multiple web services. The support of WSDL operations is intended for the design of Business Collaborations in which one or more of the business partners are not capable of supporting ebXML interchanges. The mapping provides the capability to map request, possible responses and signals to abstract operation messages. The reference to an actual WSDL file is specified as part of the Collaboration Profile Agreement (such as namespace references).

The correlation between the different operation invocations is implemented at run-time. The specification does not provide any design-time correlation specification but recommends the use of run-time correlation and endpoint references based on emerging addressing mechanisms such as WS-Addressing, WS-MessageDelivery or others.

Correlation can provide additional functionality that could be desired where complex composed activities occur, and visibility of the parties and their activities must be managed.

> ***Implementation note***
>
> *The possible capabilities of the underlying infrastructure and services chosen may impact the capability to support business requirements defined by the involved parties. For example, specific constraints may apply to WSDL-based exchanges that may not exist for those implementations using ebXML Messaging Service.*

2.4.5 Relationship to Registry/Repository

Although independent, the ebXML components are designed to work together in a loosely coupled fashion. At a minimum, the ebXML Registry/Repository could allow the discovery and use of ebBP instances. If artifacts are given a classification, the instances and the profiles of the BT patterns could be part of a business process catalogue. They may be available to an industry group, enterprise or entity. The ebXML Registry/Repository provides the capability to version and manage such artifacts (See preceding figure and a similar one in Section 3).

3 Language Overview

The ebBP technical specification defines a standard language for business process specification. An ebBP definition works with the ebXML CPP/CPA specification to bridge the gap between Business Process Modeling and the configuration of eBusiness software (See following figure). The software component that manages Business Collaborations on behalf of one business partner is referenced in this specification as the BSI. A detailed discussion on the BSI can be found in the appendices to this technical specification. The BSI supports predictable eBusiness interactions. However, this does not specifically limit the use of ebBP technical specification to those interactions. This technical specification supports the computable and executable language used for Business Collaboration, rather than the processing accomplished from the view of a single party. Predictability is supported within the scope of and at the level of abstraction that a Business Collaboration operates. The functions are described in this technical specification.

A business process specification may be used to guide other executable process mechanisms to drive enterprise components where Business Collaboration definition enables monitoring and/or control (rather than creation) of service behavior.

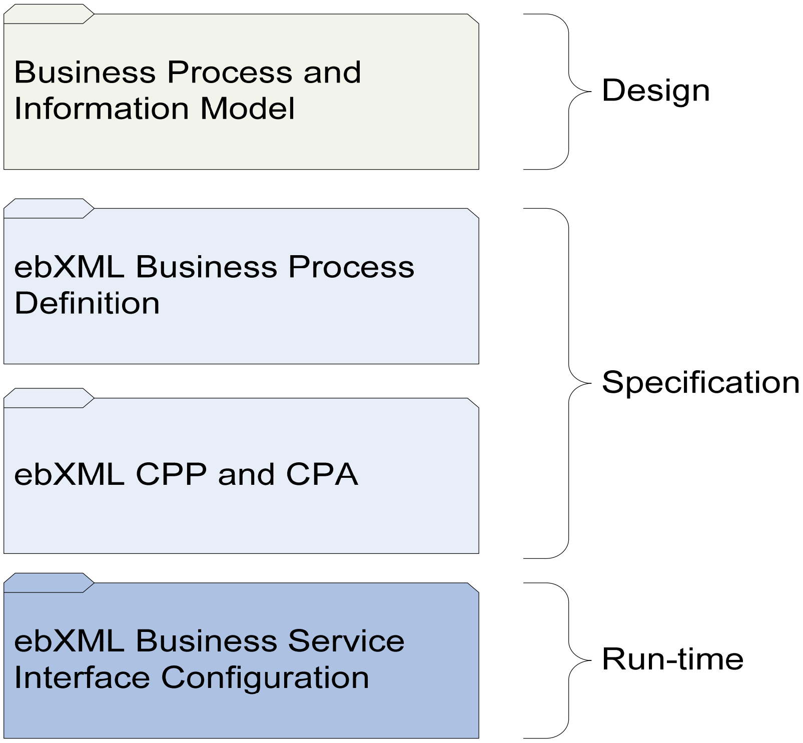

Business Process and Information Model
Design
ebXML Business Process Definition
ebXML CPP and CPA
Specification
ebXML Business Service Interface Configuration
Run-time

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 model and using the ebBP technical specification the user will then extract and format the nominal set of elements necessary to configure an ebXML runtime system in order to execute a set of ebXML Business Transactions. The result is an ebBP definition.

Alternatively the ebBP definition MAY be created directly, without prior explicit business process modeling.

An ebBP definition contains the specification of Business Transactions and the choreography of these Business Transactions that are included in Business Collaborations.

This ebBP definition may then be 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 BSI software component.

Implementation Note:

When a reference is generically made to a “BSI”, it may logically represent middleware, applications, backend systems, software or services. These components may exist within a logical enterprise (one or more domains of control). The BSI was a key component in the original ebXML framework.

The BSI represents an important component in realizing eBusiness automation and deployment. The BSI MAY be configured from an ebBP definition and a CPA. The architecture of the ebBP technical specification consists of the following functional components:

- A representation of Business Collaboration using accepted business process modeling techniques. Representations in this specification use the Business Process Modeling Notation (BPMN).
- XML Schema definition of the ebBP definition. Each ebBP definition MUST conform to this schema definition.
- Business Signal Definitions

Together these components allow you to specify the run time aspects of a business process model within the scope of this current version of the ebBP . However, all the parameters of the ebBP definition are intended to be specified at design time rather than specified or inferred at run-time. However, some values may be acquired or quantified at other than design time.

These components are shown in Figure 3 that follows.

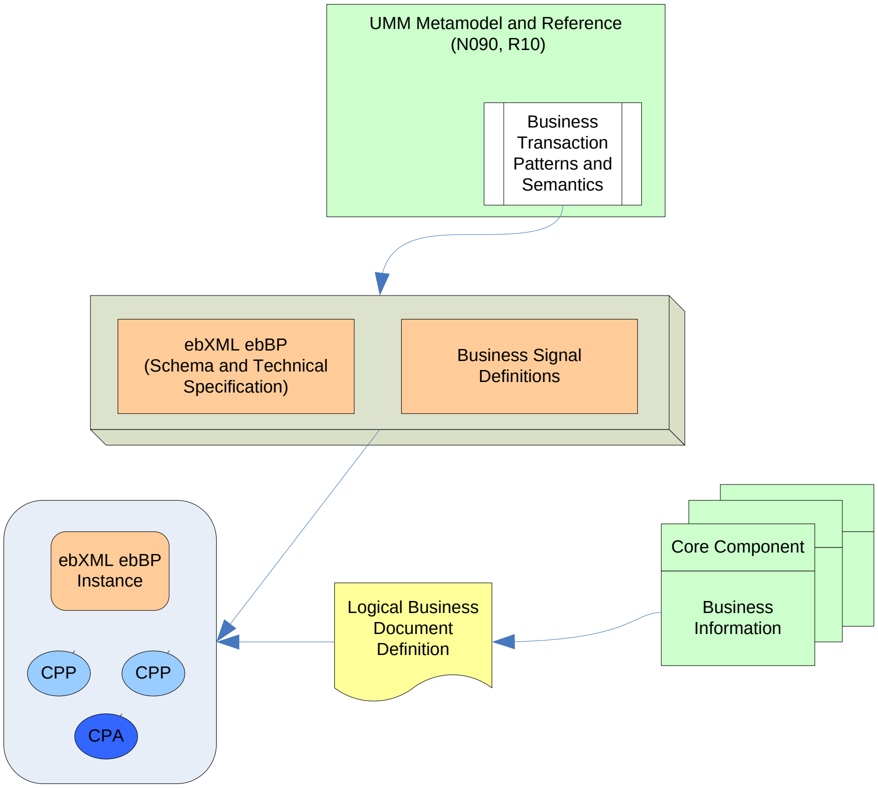

UMM Metamodel and Reference (N090, R10)
Business Transaction Patterns and Semantics
ebXML ebBP (Schema and Technical Specification)
Business Signal Definitions
ebXML ebBP Instance
CPP
CPP
CPA
Logical Business Document Definition
Core Component
Business Information

Figure 3: Relationship of ebBP technical specification to UMM, CPP/CPA and Core Components

Implementation Note:

Throughout this document, typically business partner is used when related to agreement between parties. Trading partner is used when related to CPA. Party is typically used when related to a role that a business partner plays, such as a responding party.

3.1 XML Schema Representation of Business Process Definitions

The corresponding XML Schema representation of the ebBP technical specification provides the specification for XML based definitions of an ebBP schema, and MAY serve as a target for production rules from other representations. Thus, a user MAY either create an ebBP definition directly as an XML document or from other representations.

Any methodologies and/or metamodels used for the creation of ebBP definitions MUST at a minimum support the production of the elements and relationships contained in the XML representation of the ebBP technical specification and defined in the ebBP schema. Well-formedness rules are specified in order to facilitate the understanding and use of the XML schema representation of the ebBP technical specification.

The complete XML schemas (core and signal) and their association documentation are provided in separate Schema and Signal Schema packages. Example XML instances are provided in a non-normative package outside of this technical specification and the appendices to aid user communities.

3.2 Business Signal Definitions

A Business Signal is an object that is transmitted back to the activity that initiated the transfer of execution control. Business signals have a specific business purpose and are separate from lower protocol and transport signals as defined in the ebXML Message Service Specification. The state of a given Business Transaction Activity (BTA) instance can be explicitly calculated at run-time by evaluating these signals. As such they are instrumental in establishing a Business Collaboration protocol that insures that the representation of the state of a Business Collaboration instance for each party, is strictly identical.   For example, an Acceptance Acknowledgement signal is generated after an application, service or middleware[4] has successfully processed and business validated a Business Document. The process of exchanging signals and state changes of a Business Transaction enables "state alignment" between the parties involved. 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. The Signal schema is in the packages that support this technical specification.

The ebBP technical specification provides both the structure and choreography of Business Signals. The ebXML Message Service Specification provides a reliable messaging infrastructure. This is the basis upon which the ebBP technical specification builds its protocol for business state alignment using Business Signals. The Business Signal payload structures are optional and normative and are intended to provide business semantics to the Business Signals.

A schema is provided for the possible Business Signals. Examples of sample signal instances are available in addition to this technical specification and the appendices. They may be found on the OASIS web site in a non-normative example package.

3.3 Well-Formedness Rules

A starting set of well-formedness rules is provided to aid implementers in using ebBP technical specification constructs. In Section 3.8, well-formedness rules exist for the use of, at a minimum:

- Business Collaborations
- Time To Perform
- Notification of Failure and exceptions
- Condition expressions and variables
- Web services operations
- Packages and includes

Referential and functional constraints are described in Section 3.8. Other well-formedness rules will be defined as more industry and user community knowledge and requirements are available.

[4] When a reference is generically made to an "application", it may represent middleware, applications, backend systems, software or services. These components typically exist within a logical enterprise (one or more domains of control).

3.4 Key Concepts of This Technical Specification

The ebBP specification specifies the structure and semantics of machine processable Business Collaborations definitions. These semantics are aligned with guiding principles relevant to business processes such as the UMM.

At a high level, a Business Collaboration consists of a set of roles collaborating through a set of choreographed Business Transactions by exchanging Business Documents.

These basic semantics of a Business Collaboration are illustrated in Figure 4.

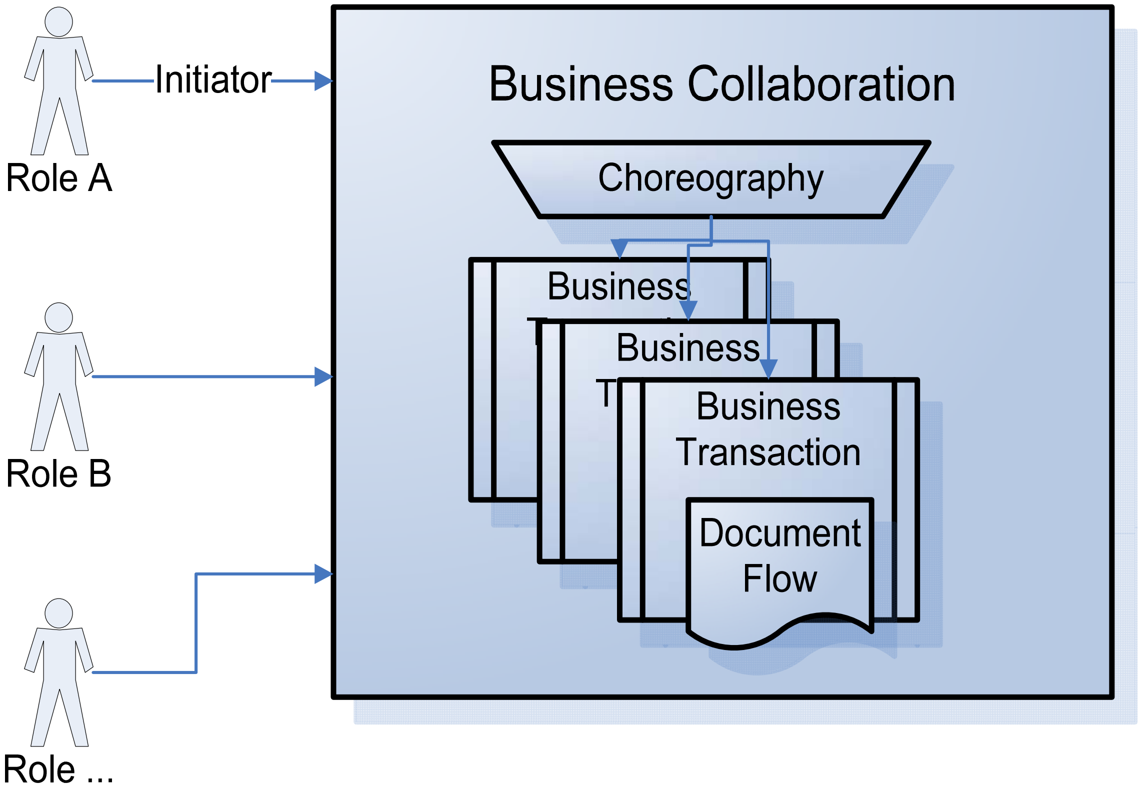


Figure 4: Illustration of the basic semantics of a Business Collaboration

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 Collaboration is defined as a choreography of Business Transactions performed relative to each other.

The following section describes the concepts of a Business Collaboration, a Business Transaction, a Business Document Flow, and Choreography. Business messages and Business Signals are discussed throughout. A business message is typically associated with a Business Document Flow rather than a Business Signal.

3.4.1 Business Collaborations

A Business Collaboration is a set of Business Activities executing Business Transactions between business partners or collaborating parties. Each business partner plays one or more abstract partner roles in the Business Collaboration. The state of the Business Collaboration is logical between the parties interacting in a peer-to-peer rather than a controlled environment. The virtual state of the Business Collaboration lies with the involved partners. Peer-to-peer collaboration may involve business partners as well as distributed collaborating parties. For the latter, one example may be cross-organizational collaboration between parties involved in technical publishing where the nested, complex activities may be required to support an authoring process. Cross-organizational collaboration may occur in many organizations, such as those government departments and agencies enabling eGovernment. The relevance of and use of the business transaction patterns in such an environment is discussed in the book by Robert Glushko and Tim McGrath, Document Engineering - Analyzing and Designing Documents for Business Informatics and Web Services.[5]

The ebBP technical specification supports several levels of Business Collaborations. Business Collaborations can be specialized as Binary or Multiparty (Business) Collaborations.[6]

When a Business Collaboration is specialized, a Binary (Business) Collaboration involves two top-level or abstract partner roles only. For the purposes of this specification, these roles are sometimes called abstract partner roles. Multiparty (Business) collaborations involve more than two abstract partner roles. Business Collaborations are expressed as a set of Business Activities between these roles. Each abstract partner role occupies a specific role when associated with a Business Activity.

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 Business Collaboration such as a Binary (Business) Collaboration within another Binary (Business) Collaboration). An example of the former is the activity of “process purchase order”. An example of the latter is the activity of “negotiating a technical contract”. The example instances, found on the OASIS web site show how an ebBP definition could be used for CPA negotiation. In either case the activities can be choreographed relative to other activities as per below.

The ability of a Binary (Business) Collaboration to have activities that in effect are executing others is the key to recursive compositions and re-use of Business Collaborations.

In essence each Business Collaboration is a re-useable protocol between two or more agreeable parties that may assume and occupy different roles at various steps in the process. Typically, a Business Transaction is defined once. However, the BT could appear many times as different Business Transaction Activities, where the roles change within the same Binary (Business) Collaboration, such as for an Offer and Counter Offer. As shown in the CPA example in the non-normative examples package, this is a known case in CPA negotiation. An activity, whether it is a Business Transaction Activity (BTA) or a Collaboration Activity represents the usage of a definition within another Business Collaboration. In the Business Transaction Activities, the abstract role in the Business Transaction becomes a specific role, where roles may change within the same Binary (Business) Collaboration. In that case, either abstract role in the Business Transaction MAY assume the initiating role in the BTA.

Business Collaboration between more than two abstract partner roles (i.e. Multiparty Collaboration) may be conducted in many presumed ways, including using coordination or as a community of peers. Functions to support Multiparty Collaboration may include status visibility, state alignment, identity, business constraints, etc. Business requirements are being gathered to gain more understanding of and define constructs for complementary functionality to support this type of Business Collaboration in addition to capabilities in this technical specification.

[5] In Chapters 9 and 10 (particularly Sections 9.3 and 9.3.1), many core aspects in ebBP are described such as the relevance of logical business documents, business transaction patterns, and context where used. As well, it outlines the importance of collaboration and the underlying patterns composed and used for business partners and collaborating parties. See: http://www.docengineering.com/.

[6] Note: In this version, specific Binary and Multiparty Collaboration elements are being retained but are to be replaced by Business Collaboration. For consistency herein, when either is referenced “(Business)” is also specified to familiarize the user community with this upcoming change.

3.4.2 Business Transactions

A Business Transaction represents an atomic unit of work that may be associated with a trading arrangement between two business partners. The scope of the ebBP technical specification is to articulate more fully the Business Transactions, rather than primarily focusing on their relationship to trading arrangements between business partners. In the future, more requirements are anticipated to further express this relationship, such as from UN/CEFACT. Atomicity in the context of this technical specification is outlined in the glossary at the end of this document.

A Business Transaction is conducted between two parties playing opposite abstract roles in that transaction. Each party, as an abstract partner, assumes an abstract role in a Business Transaction. Those roles are always generic and labeled as Requesting and Responding roles. The specific roles (e.g. buyer, seller) MUST be specified at the Business Transaction Activity level, when the Business Transaction definition is used for a distinct purpose. At that point, the abstract partner assumes and occupies a specific role, as a role occupant. Only two role occupants may be active at one time in a BTA.

Like a Binary (Business) Collaboration, a Business Transaction is a re-useable protocol between two abstract roles (explicit generic Requesting and Responding Roles). The way it is re-used is by referencing it from a Binary (Business) Collaboration through the use of a BTA as per above. In a Business Transaction Activity the specific roles of the Binary (Business) Collaboration are assigned to the execution of the Business Transaction. As indicated in the previous section, a Business Collaboration may be composed within another Business Collaboration via a Collaboration Activity. Each abstract partner participates in the Business Collaboration and occupies different role (occupants) in the included Business Transactions. How the external role in a Business Collaboration maps to the roles defined within the enclosed Business Transactions is mapped to a series of role relationships. How this is accomplished using the Performs element and external role mapping is found later in Sections 3.4.5 (shows Multiparty interactions) and 3.4.10.1.

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

A Business Transaction is a very specialized and very constrained protocol used to achieve very precise and support enforceable transaction semantics and achieve state alignment when needed between both parties. The software component managing the Business Transaction, i.e. a BSI component, SHOULD enforce these semantics. For example, the BSI monitors the timers and requirements of the Business Collaboration. It is important to note that the BSI MAY interact with other software components that check the validity of business messages or documents or perform other monitoring or application functions. A Business Transaction MUST succeed or fail from both a technical and business protocol perspective. If it succeeds from both perspectives it MAY be designated as having shared intent between the two business partners, or otherwise govern their collaborative activity. As defined by the parties' expectations, if it fails then it is null and void, and each partner MUST terminate and release any shared statement established by the transaction[7]. In addition, if it fails from protocol perspective, each party MUST synchronize their state to the state prior to the start of the transaction. For instance, a purchase order state should advance to "sent" when and only when the BSI reports a Protocol Success. In the case of a Business Failure, the state has already been "synchronized" and it is the duty of each application or service to take the proper actions. A Business Failure is any Failure that is identified by an application or service during the processing of the Business Document(s) and based on information not available in or part of the ebBP instance. 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. Success and failure, the

[7] Reference Section 3.4.9.7 for additional explanation including references to the eCommerce Patterns.

conditions and guards defined, and their relationship to Business Document Flows and Business Signals is detailed later in Section 3 (particularly Section 3.6.3).

The Business Transaction is defined as an abstract super class. It is associated with the six concrete Business Transaction patterns defined in the UMM:

- Commercial Transaction
- Information Distribution
- Notification: Note, the Notification of Failure business transaction is based on the Notification pattern.
- Query Response
- Request Confirm
- Request Response

These patterns are the semantic guidance of the Business Transaction itself. A relationship exists between the format/requirements of the pattern and the semantics of each concrete Business Transaction pattern (that map to those in the UMM). Operational semantics and other criteria apply to these patterns. Where specified in a separate contract or agreement, any of these patterns may be intentional,[8] and provide the basis of any obligation to yield accurate information.

Agreements or other business requirements may guide or change the criteria surrounding any interaction between business partners, which correspondingly influences the technologies used (such as that defined in a BSI or MSI). In essence, the guidance could result in a profile of the criteria selections of the defined pattern of the involved parties. Where the agreements actually change the baseline assumptions of these patterns, this could result in a partner-specific pattern and a subsequent profile. This is discussed in further detail in Section 3.4.9.1.

3.4.3 Business Document Flows

A Business Transaction is realized as Business Document Flows between the Requesting and Responding parties performing roles. There is always a logical Requesting Business Document, and optionally a logical Responding Business Document, depending on the desired Business Transaction configuration: e.g. one-way notification (not Notification of Failure) or information vs. two-way conversation.

The actual Business Document definition MAY be achieved using the ebXML CCTS and other related specifications. This may also be achieved by some methodology external to ebXML such as OASIS Content Assembly Mechanism (CAM). The specific context, format or other business requirements may require different approaches to provide the schema definitions (XSD or DTD) used for message exchange and which an ebBP definition can logically reference.

3.4.4 Choreography

The Choreography of a Business Collaboration describes the ordering and transitions between Business Transactions or sub collaborations within a Business Collaboration. For example, in a UML tool this could be represented with a UML activity diagram. Actually, the choreography can be specified in the ebBP schema using activity diagram concepts such as: start state, completion state, activities, Forks, Joins, decisions, transitions between activities, and guards on the transitions. It can also be specified visually in other notations such as the BPMN. However, it is beyond the scope of this document to dictate or specify which notation is used to represent a Business Collaboration.

[8] The hasLegalIntent attribute is defined later in Section 3.

3.4.5 How to Design Business Collaborations and Business Transactions

This section describes the this specification by building a complete Multiparty (Business) Collaboration ebBP 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 (Business) Collaboration re-using the Business Transaction
4. Specify a Choreography for the Binary (Business) Collaboration
5. Specify a higher level Binary (Business) Collaboration re-using the lower level Binary (Business) Collaboration
6. Specify a Multiparty (Business) Collaboration

Although this section, for purposes of introduction, discusses the specification of collaboration from the bottom up, the ebBP technical specification 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 arbitrarily complex Multiparty Collaborations. However, an ebBP definition MAY be as simple as a single Binary (Business) Collaboration referencing a single Business Transaction as part of a single BTA. This involves steps 1-3 above. Note, the ebBP technical specification does not specify any Business Process modeling methodology nor does it require the use of such methodology. A business process specification may be modeled in the BPMN or Unified Modeling Language™ (UML™)[9] activity diagrams, for example.

The example shows a “drop ship”, which involves a customer, a retailer, a vendor, and a credit authority. The order is placed by the customer and fulfilled by the vendor. The credit authority makes sure that payments are made to appropriate creditors. In the scenario, the credit authority is only capable of supporting Web Services. The standard BPMN is used for the diagrams to give a pictorial representation of this Multiparty Collaboration. The BPMN (notation) provide businesses with the capability of defining and understanding their internal and external business procedures through a Business Process Diagram, which will give organizations the ability to communicate these procedures in a standard manner. BPMN is focused on business process modeling for business analysts, using key transaction, task, activity, and pool constructs known by such experts. The use of this notation is non-normative and described in the referenced in the adjoining footnote.

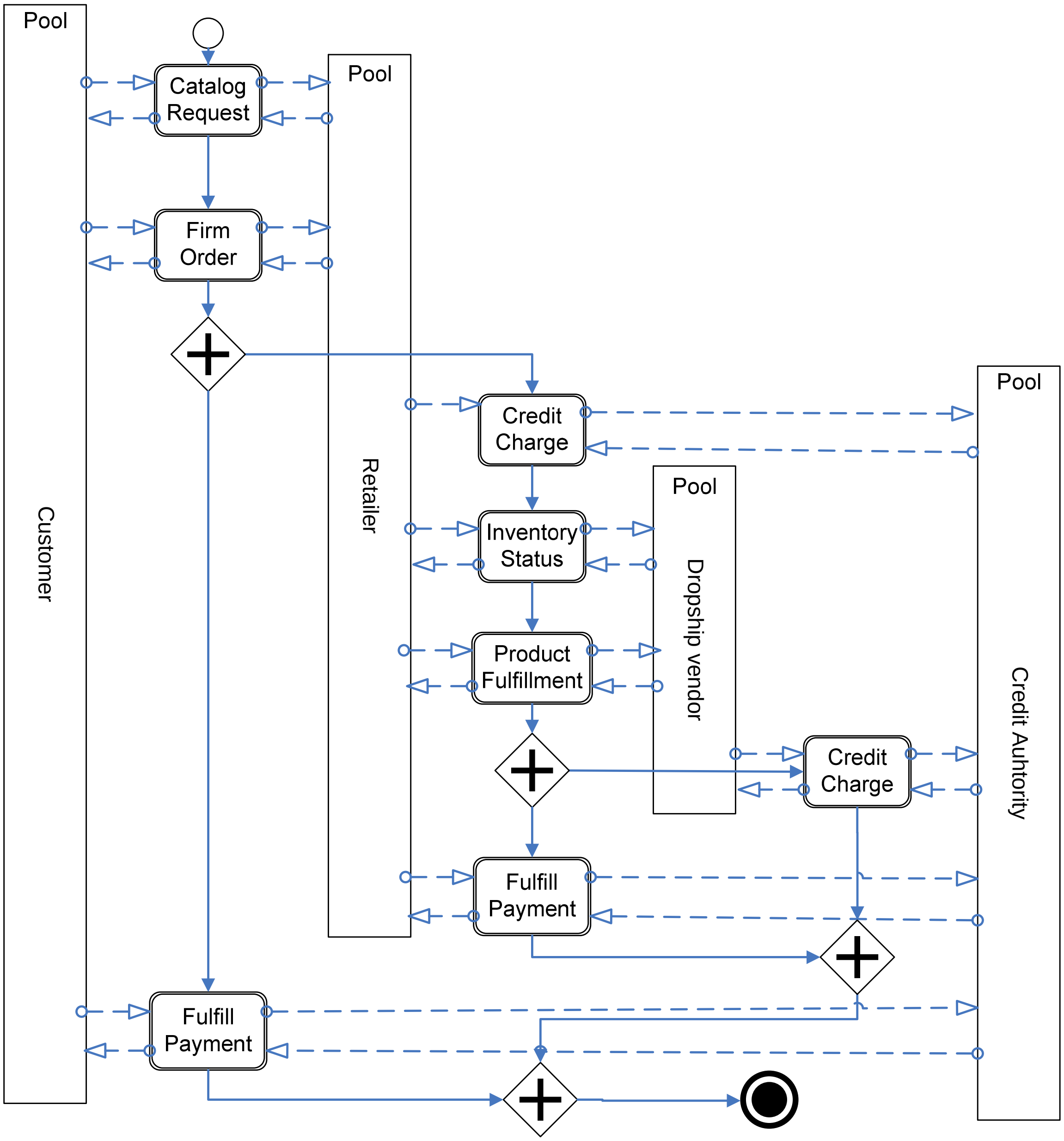
The following figure represents the overall Multiparty Collaboration using BPMN[10]. A new notation construct, a Joint Activity, is under consideration (but not yet complete) by the BPMN team at the Object Management Group (OMG). Therefore, the diagrams herein have extended BPMN to integrate that anticipated construct. In addition, comments have been received on the BPMN v1.0 specification related to message and sequence flows, and underlying semantics, and may be subject to change. The use of such flows could also change given the inclusion of collaboration constructs and support their intended use in an ebBP process definition context.

In a high-level ebBP Business Process Diagram (BPMN terminology for this visual representation), many of the BPMN constructs are used including Pool, Gateway, Sequence Flow, Message Flow, Activity, and Data Object in addition to Joint Activity. For Business Collaboration, there may be other notation constructs or semantics recommended or required. As of the date of this technical specification, these characteristics indicative of Business

[9] Object Management Group (OMG), www.omg.org.

[10] BPMN, Business Process Management Initiative, www.bpmi.org, has merged with OMG. BPMN: (www.bpmn.org) The BPMN v1.0 is an adopted OMG specification.

856 Collaboration are being discussed between the two teams, and considered for integration in an
857 incremental update to BPMN v1.0 or a future specification in OMG.



858

859

860 **Figure 5: Representation of the "DropShip" Multiparty Collaboration with a BPMN diagram**

861 All Binary (Business) Collaboration in the example feature only one BTA except two of them:
862 Credit Charge and Product Fulfillment. They are represented on the following figure using the
863 same convention.

864

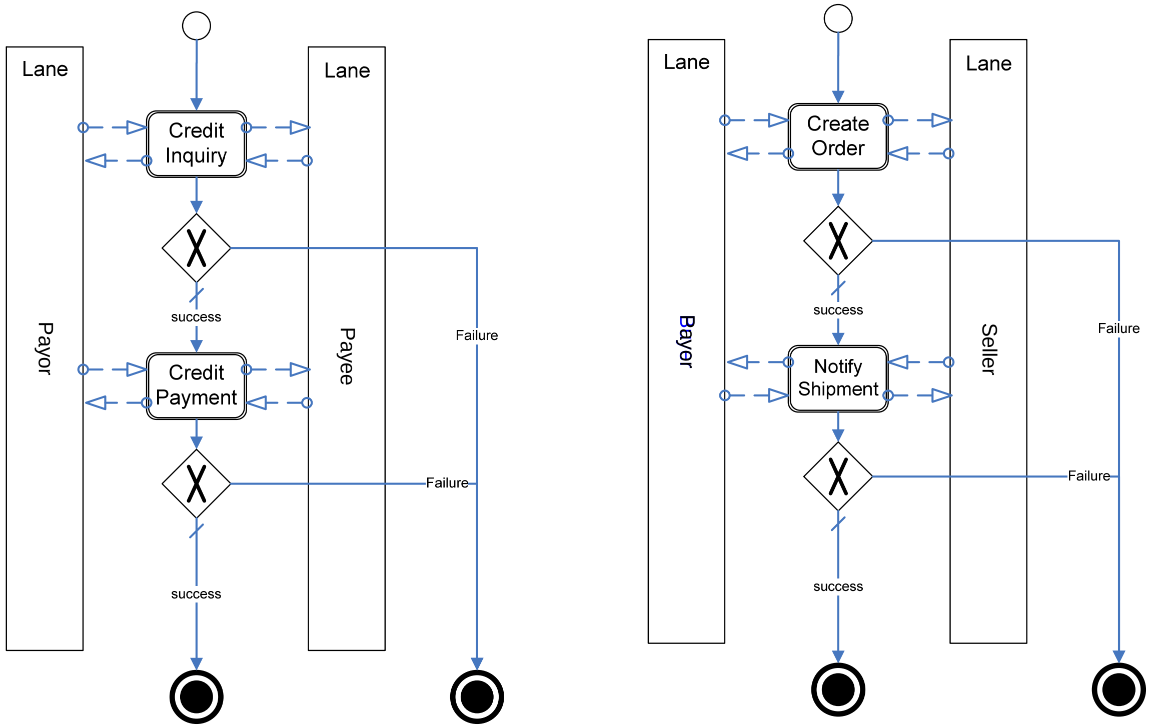


Figure 6: Representation of the “CreditCharge” and “ProductFulfillment” Binary (Business) Collaborations

3.4.6 Packages, Includes and Specifications

3.4.6.1 Packages

All elements of this specification are defined within the context of a package. Packages may contain other packages, therefore defining a hierarchy of packages. A package supports the inclusion and substitution mechanisms in the ebBP schema. Packages can be included in and reused by multiple ebBP instances.

A package defines the namespace of the elements inside it. Two model elements, such as sub-packages, with the same name within the same package MUST NOT be allowed. Two packages cannot have the same name. Model element names may be qualified.

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. Use of include requires that all model elements from the Billing package be fully qualified. So in order 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.

The use of XInclude provides an ebBP definition with an assembly mechanism that points to a URL that specifies a location from where the specification can be retrieved. It MUST point to a an existing resource. More details on the use of XInclude may be found later in this section.

3.4.6.2 Specification element

A Specification element provides the type, location, target namespace and identifiers of the specified elements. If the logical Business Document uses different namespaces, each of which has a schema, any or all may be specified using a sequence of Specification elements. For example, the retail industry uses a logical Business Document and requires different parts be identifiable (i.e. multiple references to the content structure exist which may include multiple schemas and/or namespaces). The specificationVersion may be “2” while the actual (current) artifact document version is “2.0.4”.

It is relevant to note that the ebBP technical specification focuses on the logical Business Document not a wire format. The goal was to keep logical separation of functions between implementation and the processes described. The logical business document is a semantic document. It describes the semantic content and purpose of a physical document and also may include the semantic business objective. For example, a physical Purchase Order Response document may be mapped to two or more logical documents in ebBP, "AcceptPOResponse" / "RejectPOResponse" or "ShipImmediatePOResponse" / "HoldForReleasePOResponse". The logical business document drives the business process. This allows the flexibility to describe and use semantic information from a business perspective while remaining agnostic to what happens at transport level in order to move through a series of states given the transfer of a business document.

Business documents also convey states. The ebBP process definition can provide a semantic view of how the semantic document type, its state and key elements can be used to drive the business process. This logical view maintains the value of the business process and its underlying business collaboration states. In addition to use of variables on condition expressions that are semantic element declarations (see Section 3.4.11.1.1) that drive the process, an external document reference is available in the Specification element, called externalDocumentDefRef. An example of its use could be, a local government may have variability in how procurements occurs. Using the externalDocumentDefRef (in addition to other Specification detail), that entity may need to point to third-party information to provide additional detail to control the use of that business document. This functionality is particularly relevant for user communities interested in using such as Universal Business Language (UBL) , UBL SBS or high technology trading domains.

The logical business document also provides a DocumentSpecificationType that points to more information about that specification. This capability also may assist in providing a hint to a system, while also allowing an application, middleware or a service, to bound what it may be capable of processing. An ebBP implementation MAY use DocumentSpecificationType element to point to implementation specific details that it is capable of processing.

For example, several user communities are or anticipate using a small business UBL subset, the use of a hint could enable an iterative step to automate their processes and provide flexibility in the use of context or semantic conditions understood by those groups. In this scenario, the use of 'other' enumeration value for the DocumentSpecificationType allows the integration of a human decision into a process (alert). The message exchange at the transport level and as defined in the CPA, resolve down to physical Business Documents. In addition, by user community request, 'schematron' has been added as an enumeration value to assist in providing a pointer to validation capabilities.

3.4.6.3 Include elements

If needed, only package elements MAY be included in an ebBP instance document. One or more package elements (such as elements from other ebBP instances) MAY be included using the XInclude include element. A document referenced by an include element MUST be inserted before schema or DTD validation is attempted.

```
<ProcessSpecification
xmlns="http://docs.oasis-open.org/ebxml-bp/ebbp-2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xi="http://www.w3.org/2001/XInclude"
     name="PurchasingCluster" nameID="PC23"
    uuid="urn:purchasingcluster" specificationVersion="2"
    instanceVersion="1">
    <xi:include href="signals-package-2.0.4.xml" parse="xml"
     xpointer="element(/1/1)"/>
   <BusinessDocument name="Invoice" nameID="bd-invoice">
<!--Shows use of externalDocumentDefRef optional attribute-->
     <Specification type="schema"
        location="ubl-1-0-SBS-cs/xpaths/xml/XPath/Invoice-XPath.xml"
        targetNamespace="urn:oasis:names:specification:ubl:schema:xsd:Invoice-1.0"
         name="Invoice" nameID="invoice32"
         externalDocumentDefRef="urn:oasis:names:tc:ubl:xpath:Invoice-1.0:sbs-1.0"/>
   </BusinessDocument>
   <BusinessDocument name="InvoiceResponse"
    nameID="bd-invoiceResponse">
     <Specification type="schema"
        location="http://purchasingcluster.com/InvoiceResponse.xsd"
         name="InvoiceResponse" nameID="invoice33"/>
   </BusinessDocument>
   <DataExchange name="Data:Invoice" nameID="data-invoice">
     <RequestingRole name="DIinitiator" nameID="DIinitiator1"/>
     <RespondingRole name="DIresponder" nameID="DIresponder1"/>
     <RequestingBusinessActivity name="ReqBA:SendInvoice"
        nameID="debareq-invoice"
         timeToAcknowledgeReceipt="PT6H"
        timeToAcknowledgeAcceptance="PT12H">
       <DocumentEnvelope name="DE:ProcessInvoice"
        nameID="data-de-invoice" businessDocumentRef="bd-invoice"/>
     </RequestingBusinessActivity>
     <RespondingBusinessActivity name="ResBA:ReceiveInvoice"
        nameID="debares-invoice">
       <DocumentEnvelope name="DE:ProcessInvoiceResponse"
        nameID="data-de-invoiceResponse"
        businessDocumentRef="bd-invoiceResponse"/>
     </RespondingBusinessActivity>
   </DataExchange>
   <BusinessTransaction name="BT:Invoice" nameID="bt-invoice">
     <RequestingRole name="INinitiator" nameID="INinitiator1"/>
     <RespondingRole name="INresponder" nameID="INresponder1"/>
     <RequestingBusinessActivity name="ReqBA:SendInvoice"
        nameID="reqba-invoice"
        timeToAcknowledgeReceipt="PT6H"
        timeToAcknowledgeAcceptance="PT12H">
       <DocumentEnvelope name="DE:ProcessInvoice"
        nameID="bt-de-invoice" businessDocumentRef="bd-invoice"/>
       <ReceiptAcknowledgement name="sira" nameID="sira1"
        signalDefinitionRef="ra2"/>
       <ReceiptAcknowledgementException name="sirae"
        nameID="sirae1" signalDefinitionRef="rae2"/>
     </RequestingBusinessActivity>
     <RespondingBusinessActivity name="ResBA:ReceiveInvoice"
```

```
       nameID="resba-invoice">
      <DocumentEnvelope name="DE:ProcessInvoiceResponse"
       nameID="bt-de-invoiceResponse"
       businessDocumentRef="bd-invoiceResponse"/>
      <ReceiptAcknowledgement name="sira" nameID="sira2"
       signalDefinitionRef="ra2"/>
      <ReceiptAcknowledgementException name="sirae"
       nameID="sirae2" signalDefinitionRef="rae2"/>
    </RespondingBusinessActivity>
  </BusinessTransaction>
</ProcessSpecification>
```

In this example, Signals-package-2.0.4.xml is the target xml document that will be parsed as xml and whose first child Package element of the ProcessSpecification element will be inserted. In this example the XInclude reference will resolve the ra2 and rae2 signal references.

See the http://www.w3.org/2001/XInclude namespace. Implementers MUST ensure that attribute values of nameID are unique (i.e. no collisions occur). ebBP implementations MUST process the XInclude include element by making the appropriate insertions prior to schema or DTD validation is attempted. The XInclude mechanism replaces the include element in previous versions of ebXML BPSS.

If a package has a parent, the parentREF will enable inclusion all elements in the package’s hierarchy or tree. Then, an implementer MAY be capable of recreating a tree without relying on package names.

Arbitrary or invalid construction using XInclude is not recommended. In this technical specification, the effective use of XInclude SHOULD be restricted to inclusion of packages only (that may include other packages). This simple approach facilitates the use of this mechanism to support composition of ebBP definitions.

3.4.7 Versioning

The ebBP technical specification supports versioning of an ebBP instance with instanceVersion attribute of ProcessSpecification element. The instanceVersion attribute MAY be used to distinguish different revisions of a business process. The ebBP technical specification does not define specific format for the value of instanceVersion attribute. Authors, such as those within an industry, MAY choose arbitrary text of their convenience to recognize their assigned instanceVersion.

The instanceVersion attribute should be differentiated from the specificationVersion attribute, which is the major version identifier of ebBP technical specification of which that ebBP instance MUST conform. In this case, specificationVersion MUST always have value "2", if specified, for ebBP instances that conform to this major version of the technical specification. Two process models with different specification versions could in principle have the same instance version. The ebBP schema version MUST be defined by namespace (where minor variant versions within a namespace are handled by different URLs for specific schema location). The namespace URL always contains the most up-to-date schema. For example, the ebbp-2.0.4.xsd (ebBP schema document version for artifact name) [minor (“0”) and release (“2”)] resides in the v2.0 namespace (i.e. …ebbp-2.0 namespace and specificationVersion = 2) [major].

The attribute uuid MUST NOT be used for the purpose of versioning, so that even a change introduced by AttributeSubstitution (to Business Documents’ schemas, for example), would be marked by a new uuid. So while the same instance version could appear in two process documents with different schema namespaces, for example, they each would have different uuids. The uuid is not a guarantee that the version is the same. Take two examples, one that is more predictable. In the first case, the uuid is the same for different business process definitions. Therefore, they are the same version (ebBP schema and, where used, instance and specification version). However, in a second case: If the definitions exist in different repositories, each could have a different uuid. In implementation, tools (such as modeling tools) MAY use the uuid attribute value as a direct pointer to a particular ebBP instance within a namespace of a repository.

```
<ProcessSpecification xmlns="http://docs.oasis-open.org/ebxml-bp/ebbp-2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:xi="http://www.w3.org/2001/XInclude"
    xsi:schemaLocation="http://docs.oasis-open.org/ebxml-bp/ebbp-2.0 http://docs.oasis-open.org/ebxml-bp/ebbp-
2.0”
    name="PurchasingCluster"
    nameID="PC23"
    uuid="urn:purchasingcluster"
    specificationVersion="2"
    instanceVersion="2.1" >
```

An industry may choose to use a specific instance version such as Australian Wheat Board v2.1. The specificationVersion for the technical specification resolves to 2 (version) while its document artifact name is 2.0.4 (version).

Further explanation related to the use of NameID for referencing is detailed later in Section 3.8.

3.4.8 Attribute Substitution Sets

There is a requirement for business process specifications that are more loosely coupled to technology and business details, such as specific document formats and structures and timing parameters. An industry MAY choose to specialize it for their domain context and definition. This can allow a Business Collaboration to be bound to many Business Document requirements and formats. 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 ebBP specification and MAY replace attribute values only. As such references to Business Documents definitions (abstract or not) within a Business Transaction definition MAY be replaced with other Business Document definition references. A Substitution Set is a container for one or more AttributeSubstitution elements. The entire SubstitutionSet specifies attribute values that should be used in place of some attribute values in an existing ebBP specification.

Where used, the attribute or document value SHOULD be used in place of some value in an existing ebBP specification. Attribute substitution MAY be used for document substitution.

3.4.9 Business Transaction and Business Document Flow

3.4.9.1 Key Semantics of a Business Transaction

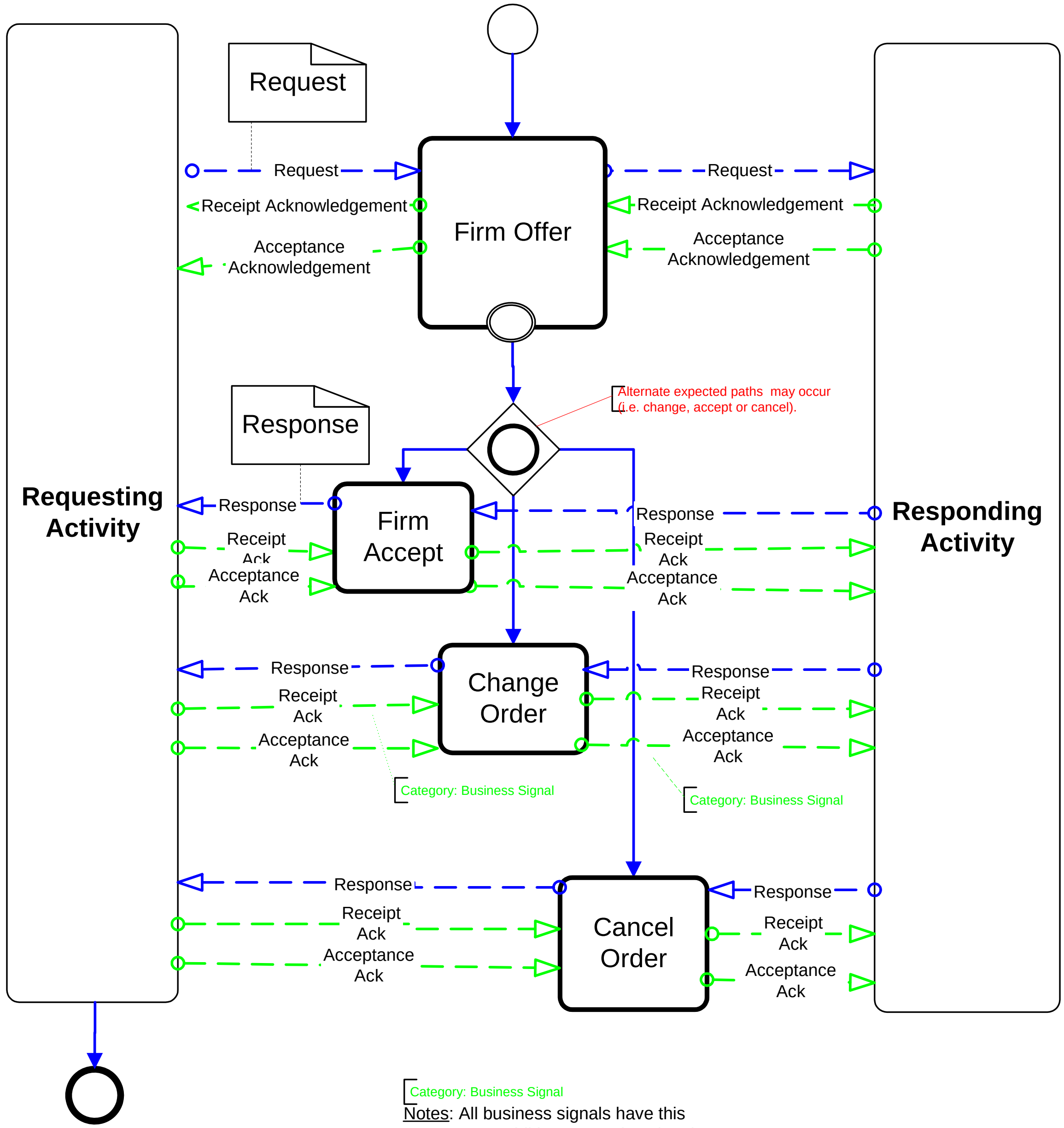
As a unit of work in a trading arrangement between two business partners, a Business Transaction consists of a Requesting Business Activity, a Responding Business Activity, and one or two Document Flows between them. A Business Transaction may involve the exchange of one or more Business Signals that govern the use and meaning of acknowledgements.

Business signals acknowledging the Document Flow may be associated with each Document Flow.

Figure 7 presents an example of Document Flows and Business Signals within a Business Transaction. This Business Transaction has been represented in BPMN. As indicated for Figure 6, the BPMN v1.0 could be extended while changes to support Business Collaboration are considered by the BPMN team in the Object Management Group (OMG). In a Business Collaboration, several possible (expected) paths of business messages exist, and the semantics of Fork and Join are also important.

Changes are under discussion by OMG BPMN team. The joint activity previously referenced is being used in anticipation of those changes.

In addition, business messages have been represented by a thicker blue message flow, while signals are green message flows. These are allowed extensions in BPMN v1.0. Currently, business signals or messages are not differentiated in a standard way in this notation.



Category: Business Signal

Notes: All business signals have this category. In addition, exception signals are not explicitly modeled for simplicity.

Figure 7: Possible Document Flows and signals and their sequence

The Requesting role performing the Requesting Business Activity and the Responding role performing the Responding Business Activity are abstract (placeholders). These roles become explicit and specific in context when the transaction is used within a BTA as part of a Business Collaboration. In the Business Transaction, the abstract roles are declared. However, there is no need to make these roles concrete such as buyer or seller. In particular some Business Transactions, for example “Cancel Purchase Order” MAY be used either way within the same Business Collaboration as two different Business Transaction Activities. In practice, roles may be implicit such as Initiator or Responder. To promote consistency and support role switching where applicable, these implicit roles of the abstract partners are explicitly declared and can be referenced in the BT. Role changes and role bindings are described in more detail in Section 3.

There is always a Request Document Flow. A Business Transaction definition specifies whether a Respond Business Document is required.

The Request Document Flow relates to the Business Transaction being implemented and may have a relationship with other Business Transactions (where applicable). For example, a Request Document Flow may be implicit or manual, or associated with a previous Business Transaction. A common example of a Request Document Flow that is a Notification Business Transaction (related to the Notification Pattern) is an Advance Ship Notice or Despatch (Dispatch) Advice. These are both requests. In this case, a previous Commercial Transaction may have been completed between two parties and one party desires to notify of shipment. That shipment may be logically considered an additional response to the original Business Transaction. However, the original Business Transaction and this Notification are separate. This and related cases are outlined in the appendices to this technical specification.

If defined within the parties’ expectations, a Business Transaction involving a response (to a request) may be associated with the formation of contracts and agreements.

A Business Action, an abstract element, is the holder of attributes that are common to both Requesting Business Activity and Responding Business Activity. This element cannot appear in ebBP instances. Irrespective of whether or not a Response Business Document is required (i.e. no DocumentEnvelope), a Responding Business Activity exists to support the mapping of the corresponding role and business action. Even when no Response Business Document is produced, there is a Responding Business Activity that occurs that receives and process the Request Business Document. Each activity has roles bound and linked to it.

A Business Transaction itself is abstract (i.e. the BusinessTransactionHead in the schema). In this version, eight overall patterns are available. There are six concrete Business Transactions patterns defined which are related to those defined by UMM and that map to Business Transactions. For this version, the ebBP technical specification has included these six concrete patterns, while retaining the original Business Transaction abstract pattern for conversions purposes only. Implementations are strongly encouraged to use the concrete Business Transactions when creating new ebBP instances. Implementations MAY use LegacyBusinessTransaction when converting instances in previous versions of ebXML BPSS.

In addition to the six concrete patterns referenced above and the LegacyBusinessTransaction, a Data Exchange pattern has also been defined to allow user communities to create a specialized pattern or extend the existing concrete ones. If a pattern is defined (outside of the concrete six ones), the pattern business semantics, underlying and surrounding protocol, state synchronization, or effects of extension are the responsibility of the defining parties. Extensibility of the concrete patterns is outside of the defined BT protocol, the Data Exchange element allows their redefinition. Outside of the syntactic parameters defined, this element allows parties to define their own operational and business semantics related to this pattern.

In addition, for v2.0.x versions, the existing pattern attribute has been retained. This pattern attribute SHOULD be used when the explicit (concrete) patterns are not used. Conversely, when a concrete pattern is used, the pattern attribute SHOULD NOT be used. The pattern name is extensible.

The six concrete patterns are summarized below. In addition, the customizable Data Exchange and historical Business Transaction (pattern) are also included for completeness.

- Commercial Transaction : For Commercial or Business Transaction, either element relates to the same Commercial Transaction BT pattern (to serve different communities to achieve a similar goal). Typically this pattern is a formal obligation between parties. Note, although specified as 'Commercial Transaction' in the UMM R10 transaction patterns, two separate elements were chose to reference back to it via the BusinessTransactionType in the ebBP v2.0.x schemas. This recognizes the use of the pattern in a broader user community and these changes were a result of community requests.

- Notification: Used for business notifications such as a Notification of Failure Business Transaction in line with a Commercial Transaction pattern. Represents a formal exchange between parties. Typically, in the case of NOF, used to render a Business Transaction as null and void. An Advance Ship Notice or Status Order are also business notifications.

- Information Distribution : Represents an informal information exchange between parties.

- Query / Response : Used by a Requester for an information query of which the responding party already has.

- Request / Confirm : Used where an initiating party requests confirmation about their status with respect to previous obligations or a Responder's business rules.

- Request / Response : Used when an initiating party requests information that a responding party already has and when the request for business information requires a complex interdependent set of results.

- Data Exchange : Allows a partner, industry or community to define a specific Business Transaction pattern not in the concrete set. The semantics used for data exchange are partner-specific.

- Legacy Business Transaction : Retained in v2.0.x technical specifications for conversion purposes only to enable the user community to migrate to the concrete patterns. This pattern is not recommended for use for the concrete Business Transaction patterns.

The patterns are applied to Business Transactions. In a Business Transaction, a Request may be manual, implicit or not apply, whereby the intent of the involved parties may be important. One such case is described in further detail in the appendices to this technical specification.

The Business Transaction patterns are described in further detail in the following matrices. Table 1 represents each pattern and their relationship to Business Signals and responses. The remaining matrices actually provide greater detail of the 6 concrete Business Transaction Patterns (excluding the partner-defined Data Exchange and LegacyBusinessTransaction conversion patterns available for use). These matrices provide relevant capabilities associated with the six concrete patterns, but do not enforce how trading partners use those capabilities. These matrices SHOULD be used. For example, the parties may select other quality of service related, operational semantics (such as isIntelligibleCheckRequired or retryCount). These are further described later in Section 3. In the succeeding tables, some usage recommendations are made such as the use of an Acceptance Acknowledgement Business Signal. The accompanying ebBP schema supports these recommendations. In some cases (i.e. where a capability is optional and other alternate capabilities may be chosen by the parties), the usage MAY to be specified by those parties. For example, isGuaranteedMessageDeliveryRequired has a default of 'false' although it is recommended to be ’true’ for most uses.

Note: Obligation herein is described as a responsibility to provide accordant information, which differs from residual obligation (obligation to a subsequent action).

Business Transaction Pattern	**Concrete or Extensible**	**Response**	**Request Receipt Ack/ Exception**	**Request Acceptance Ack/ Exception**	**Response Receipt Ack/ Exception**	**Response Acceptance Ack/ Exception**
Commercial Transaction : Business or Commercial Transaction ***(see Note 1)***	Concrete	Yes	Yes	Optionally recommended	Yes	Optionally recommended
Notification	Concrete	No	Yes	Optional	N/A	N/A
Information Distribution	Concrete	No	Optional	No	N/A	N/A
Query / Response	Concrete	Yes	Optional	No	Optional	No
Request / Confirm	Concrete	Yes	Yes	No	Yes	No
Request / Response	Concrete	Yes	Optional	No	Optional	No
Data exchange	Extensibility pattern	Optional	Optional	Optional	By agreement	By agreement
Legacy Business Transaction ***(Retained for conversion only. See below)***	Conversion use only	Yes	Yes	Yes	No	No

Table 1 Business Transaction Message Exchange Patterns

Note 1 : The Commercial Transaction or Business Transaction elements relate to the Commercial Transaction pattern via the BusinessTransactionType in the ebBP v2.0.x schemas. Both carry the same semantics and syntactic constructs, and operational criteria. Their differentiation and separation in the ebBP schema into a Commercial Transaction and Business Transaction allows similar usage by different communities.

> *Implementation Note: The Legacy Business Transaction may be used with a XSLT transform to start to migrate and upgrade to the ebBP v2.0.x schemas. Hints and a starting example (partial) are provided outside of this technical specification. Any transformation will add capabilities such as the Business Signals and criteria surrounding the use of the concrete BT patterns.*

Pattern/Criteria	**Short Description**	**Other Comments**	**Example Use Case**
Commercial Transaction *(Business Transaction Type for Commercial Transaction or Business Transaction)*	Formal obligation between parties	Can relate to use of NOF. This construct has historically and commonly known as a business transaction with the formal pattern being named Commercial Transaction.	A buyer requests a product or service in a specific time delivered to a pre-determined location from a Seller. Accepting the obligation, the Seller agrees and commits to delivery to complete a business transaction. The parties may have a pre-existing agreement to exchange goods and payment.
Notification	A formal information exchange between parties.	NOF can apply to timeout on responding party's document or an issue with the received responding party's document (signature missing or invalid, erroneous, not authorized - maps back to TPA). It is recommended this be sent over an alternate communication channel. How and when the NOF is used is TPA specific. Provides further flexibility given decisions between the parties.	A requesting role that throws a business protocol exception terminates the transaction and then sends a notification revoking the offending business document request. The requesting role cannot send a business signal to the responding role to terminate the transaction. A responding role that throws a business protocol exception signals the exception back to the requesting role and then terminates the business transaction.
Information Distribution	An informal information exchange between parties		A Seller notifies its Buyers of the release of a new product line that become part of an product catalog. As each Buyer retains a copy of the product catalog, they may acknowledge receipt. Without non-repudiation, Information Distribution may be difficult to prove authorship and adherence.
Request-Response	A request and response where no residual obligation is created (for example, a request for price and availability). The request/response activity pattern shall be used for business contracts when an initiating partner requests information that a responding partner already has and when the request for business information requires a complex interdependent set of results.	Typically no residual obligation created. Requires some business processing before the results of a query are provided.	A Buyer asks a Seller in a request for the price and availability of a particular product. This request does not result in the responding party allocating product for future purchase. The Seller queries its inventory and other applications to provide a sufficient response by checking their Supply Chain Management and Inventory systems. The Seller has to calculate the current price based on availability, its Suppliers' details, etc. Most often, the Request-Response does not involve a simple Yes/No answer from the responding party.

Pattern/Criteria	Short Description	Other Comments	Example Use Case
Request-Confirm	Used for business contracts where an initiating partner requests confirmation about their status with respect to previous contracts or a responder's business rules.	Typically no residual obligation created.	A Buyer requests from a Seller if it is still authorized to sell certain product. The Buyer expects a confirmation response. A response does not equate to an obligation, although further action could subsequently occur. A previous contract may or may not have existed between the parties. The Seller confirms he is still authorized to sell the product. Typically, the Request-Confirm involves a simple Yes/No answer from the responding party.
Query Response	Used by a requester for an information query that responding partner already has.	This pattern should be used when the response meets the specified constraining criteria. If this involves a complex set of results, use request-response pattern. Use when no interdependency exists between the query results. Can use this pattern when querying business information and for specifying the structure of the response (without complex constraints).	A Buyer asks a Seller in a request for the price and availability of a particular product. This request does not result in the responding party allocating product for future purchase. The Seller maintains a online product catalog of products and can provide the Buyer a response without complex constraints or backend processing.
General Notes:			
UMM R10, Chapter 9 specifies the RA and AA on the responder to the requester. Here experts have historically differed on the use of the signals on requester to the responder.	*Note: More information may need to be derived from UMM R10, Chapter 8. In work.*	*Note: The Commercial Transaction pattern is not the legacy conversion Business Transaction pattern. The Commercial Transaction pattern in the UMM R10, Commercial Transaction is mapped to the ebBP v2.0.4 concrete pattern. That concrete pattern is typed and mapped to the ebBP Business Transaction Type that relates to (1) Commercial Transaction and (2) Business Transaction, that allows usage by different communities (commercial or not).*	
Key:	Mapping titles	Not applicable	Not allowed

Table 2 Concrete Business Transaction Pattern Descriptions and Examples

Pattern/Criteria	**Receipt Ack/Exception (on request)**	**Accept Ack/Exception (on request)**	**Receipt Ack/Exception (on response)**	**Accept Ack/Exception (on response)**	**Response**	**NOF Possible**
	Can include grammar, sequence and syntax validation.	*Includes content validation*	*Can include grammar, sequence and syntax validation.*	*Includes content validation*		
Commercial Transaction *(Business Transaction Type for Commercial Transaction or Business Transaction)*	X	Optional but strongly recommended	X	Optional but strongly recommended	X (if accepted and if substantive)	X (if control failure)
		1. If negative AA, no response is sent by the responding party. *2. If positive AA, a business response is sent by the responding party. The AA and the business response are in the same business transaction (and BT activity).* *3. Users are encouraged to review UN/ECE Recommendations 26 and 31 about business enforceability. AA allows state alignment to optimize processes accordingly.* *4. The response may fulfill the AA and the response for the party commitments. An AA is not the response.* *5. Substantial risk exists when it is not used for state alignment.*		*1. The responding party can issue an exception. The agreement may dictate the applicable conditions.* *2. Users are encouraged to review UN/ECE Recommendations 26 and 31 about business enforceability. AA allows state alignment to optimize processes accordingly.* *3. The response may fulfill the AA and the response for the party commitments. An AA is not the response.* *4. Substantial risk exists when it is not used for state alignment.*		
						Business retry may also apply.

Pattern/Criteria	**Receipt Ack/Exception (on request)**	**Accept Ack/Exception (on request)**	**Receipt Ack/Exception (on response)**	**Accept Ack/Exception (on response)**	**Response**	**NOF Possible**
Notification	X	Optional				Pattern used for NOF.
		This is a business message, and therefore, because of the intentional nature, a n AA is optional.				*An NOF may also be used in a business collaboration that includes multiple transactions. In that case, any of the patterns are used together. If a non-receipt occurs, for example, a NOF may result to set aside the Business Collaboration.*
						Business retry may also apply.
Information Distribution	Optional					Not allowed
						Business retry may also apply.
Request-Response	Optional	Not allowed explicitly	Optional	Not allowed explicitly	X	Not allowed
						Business retry may also apply.
Request-Confirm	X	Not allowed explicitly	X	Not allowed explicitly	X	Not allowed
						Business retry may also apply.

Pattern/Criteria	**Receipt Ack/Exception (on request)**	**Accept Ack/Exception (on request)**	**Receipt Ack/Exception (on response)**	**Accept Ack/Exception (on response)**	**Response**	**NOF Possible**
Query Response	Optional	Not allowed explicitly	Optional	Not allowed explicitly	X	Not allowed explicitly
	General Notes:					*Normally business retry may apply.*
	UMM R10, Chapter 9 specifies the RA and AA on the responder to the requester. Here experts have historically differed on the use of the signals on requester to the responder.	*Note: More information may need to be derived from UMM R10, Chapter 8. In work.*	*Note: The Commercial Transaction pattern is not the legacy conversion Business Transaction pattern. The Commercial Transaction pattern in the UMM R10, Commercial Transaction is mapped to the ebBP v2.0.4 concrete pattern. That concrete pattern is typed and mapped to the ebBP Business Transaction Type that relates to (1) Commercial Transaction and (2) Business Transaction, that allows usage by different communities (commercial or not).*			
	Key:	Mapping titles	Not applicable	Not allowed		

Table 3 Concrete Business Transaction Pattern Operational Semantics (1 of 4)

Pattern/Criteria	**Non-repudiation receipt (on request)**	**Non-repudiation of content and origin (on request)**	**Non-repudiation receipt (on response)**	**Non-repudiation of content and origin (on response)**
Commercial Transaction *(Business Transaction Type for Commercial Transaction or Business Transaction)*	X	X	X	X
		Includes content and origin (responding role identity) validation.	*Although it is possible one may consider non-repudiation could be optional for a receipt on a response, this is strongly recommended.*	*Includes content and origin (responding role identity) validation.*
Notification	X	X		
Information Distribution	Not allowed (no NR requirements exist)	Not allowed (no NR requirements exist)		
Request-Response	Optional	Optional	Optional	Optional
	A receipt acknowledgment is allowed and therefore, non-repudiation of receipt may apply. By agreement, the parties may determine this is an implicit input to a future decision.	*By agreement, the parties may determine this is an implicit input to a future decision.*	*A receipt acknowledgment is allowed and therefore, non-repudiation of receipt may apply. By agreement, the parties may determine this is an implicit input to a future decision.*	*By agreement, the parties may determine this is an implicit input to a future decision.*
Request-Confirm	Optional	Optional	Optional	Optional
	Request-Confirm is a pattern where non-repudiation can be changed without changing semantics.	*Requesting business document is allowed. Repudiation of content should therefore be optional but allowed.*	*Request-Confirm is a pattern where non-repudiation can be changed without changing semantics.*	*Responding business document is allowed. Repudiation of content should therefore be optional but allowed.*

Pattern/Criteria	**Non-repudiation receipt (on request)**	**Non-repudiation of content and origin (on request)**	**Non-repudiation receipt (on response)**	**Non-repudiation of content and origin (on response)**
Query Response	Optional	Optional	Optional	Optional
	Requesting business document is allowed. Repudiation of receipt should therefore be optional but allowed.	*Requesting business document is allowed. Repudiation of content should therefore be optional but allowed.*	*Responding business document is allowed. Repudiation of receipt should therefore be optional but allowed.*	*Responding business document is allowed. Repudiation of content should therefore be optional but allowed.*
	General Notes:	*UMM R10, Chapter 9 specifies the RA and AA on the responder to the requester. Here experts have historically differed on the use of the signals on requester to the responder.*	*Note: More information may need to be derived from UMM R10, Chapter 8. In work.*	*Note: The Commercial Transaction pattern is not the legacy conversion Business Transaction pattern. The Commercial Transaction pattern in the UMM R10, Commercial Transaction is mapped to the ebBP v2.0.4 concrete pattern. That concrete pattern is typed and mapped to the ebBP Business Transaction Type that relates to (1) Commercial Transaction and (2) Business Transaction, that allows usage by different communities (commercial or not).*
	Key:	Mapping titles	Not applicable	Not allowed

Table 4 Concrete Business Transaction Pattern Operational Semantics (2 of 4)

Pattern/Criteria	TTP	Time to Acknowledge Receipt	Time to Acknowledge Acceptance	Formal or Informal	Has Legal Intent
Commercial Transaction *(Business Transaction Type for Commercial Transaction or Business Transaction)*	X	X	X	Formal	default="false"
					By agreement. Typically, this attribute is consistent between Notification and Commercial Transaction/Business Transaction (Commercial Transaction pattern). In general, this pattern meets legal enforceability reqts.
Notification	X	X	Optional	Formal	default="false"
					By agreement. Typically, this attribute is consistent between Notification and Commercial Transaction/Business Transaction (Commercial Transaction pattern). In general, this pattern meets legal enforceability reqts.
Information Distribution	X	Optional	🟧	By agreement	default="false"
					By agreement
Request-Response	X	Optional	Not allowed explicitly	By agreement	default="false"
					By agreement

Pattern/Criteria	**TTP**	**Time to Acknowledge Receipt**	**Time to Acknowledge Acceptance**	**Formal or Informal**	**Has Legal Intent**
Request-Confirm	X	X		By agreement	default="false"
					By agreement
Query Response	X	Optional	Not allowed explicitly	By agreement	default="false"
					By agreement
			General Notes:		
			UMM R10, Chapter 9 specifies the RA and AA on the responder to the requester. Here experts have historically differed on the use of the signals on requester to the responder.	*Note: More information may need to be derived from UMM R10, Chapter 8. In work.*	*Note: The Commercial Transaction pattern is not the legacy conversion Business Transaction pattern. The Commercial Transaction pattern in the UMM R10, Commercial Transaction is mapped to the ebBP v2.0.4 concrete pattern. That concrete pattern is typed and mapped to the ebBP Business Transaction Type that relates to (1) Commercial Transaction and (2) Business Transaction, that allows usage by different communities (commercial or not).*

Table 5 Concrete Business Transaction Pattern Operational Semantics (3 of 4)

1237

Pattern/Criteria	**isGuaranteedMessageDeliveryRequired**	**documentSecurity (isConfidential, isTamperDetectable, isAuthenticated on Document Envelope)**
Commercial Transaction *(Business Transaction Type for Commercial Transaction or Business Transaction)*	default = 'false'	X
	Strongly recommended to support state alignment.	*If non-repudiation of content is required, the enumeration selected for each of these values should be other than 'none.' Typically, this occurs in situations where hasLegalIntent applies.*
Notification	default = 'false'	X
	Strongly recommended to support state alignment.	*If non-repudiation of content is required, the enumeration selected for each of these values should be other than 'none.' Typically, this occurs in situations where hasLegalIntent applies.*
Information Distribution	default = 'false'	Optional
	Strongly recommended to support state alignment.	
Request-Response	default = 'false'	Optional
	Strongly recommended to support state alignment.	*By agreement of the parties. Non-repudiation of content suggests that the business document will be protected as specified for the Document Envelope. Typically, this occurs in situations where hasLegalIntent applies.*
Request-Confirm	default = 'false'	Optional
	Strongly recommended to support state alignment.	*By agreement of the parties. Non-repudiation of content suggests that the business document will be protected as specified for the Document Envelope. Typically, this occurs in situations where hasLegalIntent applies.*

Pattern/Criteria	**isGuaranteedMessageDeliveryRequired**	**documentSecurity (isConfidential, isTamperDetectable, isAuthenticated on Document Envelope)**
Query Response	default = 'false'	Optional
	Strongly recommended to support state alignment.	*By agreement of the parties. Non-repudiation of content suggests that the business document will be protected as specified for the Document Envelope. Typically, this occurs in situations where hasLegalIntent applies.*
General Notes:	*UMM R10, Chapter 9 specifies the RA and AA on the responder to the requester. Here experts have historically differed on the use of the signals on requester to the responder.*	*Note: More information may need to be derived from UMM R10, Chapter 8. In work.*
Key:	Not allowed	*Note: The Commercial Transaction pattern is not the legacy conversion Business Transaction pattern. The Commercial Transaction pattern in the UMM R10, Commercial Transaction is mapped to the ebBP v2.0.4 concrete pattern. That concrete pattern is typed and mapped to the ebBP Business Transaction Type that relates to (1) Commercial Transaction and (2) Business Transaction, that allows usage by different communities (commercial or not).*
	Mapping titles	Not applicable

Table 6 Concrete Business Transaction Pattern Operational Semantics (4 of 4)

For the six concrete patterns and the LegacyBusinessTransaction (conversion only pattern) additional operational semantics may exist in the patterns matrices rather than being held in the ebBP schema. For example, manual or implicit actions by an involved party may be relevant in the ebBP process definition, particularly to provide state transition information in the Business Collaboration for monitoring. In the appendices to this technical specification, a brief description is provided about how the patterns may be used when manual or implicit actions exist. In future versions, more semantics may be defined and included in the ebBP technical specification and/or schema as business requirements are identified or user community feedback received.

3.4.9.2 Sample syntax

Here is a simple QueryResponse Business Transaction definition with just a Requesting and Response Document Flow:

```
<!--…-->
<QueryResponse name="Catalog Request" nameID="ID100" isGuaranteedDeliveryRequired="false">
        <RequestingRole name="CRinitiator" nameID="CRinitiator1"/>
        <RespondingRole name="CRresponder" nameID="CRresponder1"/>
        <RequestingBusinessActivity name="requestCatalog" nameID="ID101">
            <DocumentEnvelope name="Catalog Request" nameID="ID102" businessDocumentRef="ID1000"/>
        </RequestingBusinessActivity>
        <RespondingBusinessActivity name="sendCatalog" nameID="ID103">
            <DocumentEnvelope name="Catalog Response" nameID="ID104" isPositiveResponse="true"
                businessDocumentRef="IDs1001"/>
        </RespondingBusinessActivity>
    </QueryResponse>
<!--…-->
```

3.4.9.3 Business Signals

The type of Business Transaction specifies whether a Receipt Acknowledgement and/or an Acceptance Acknowledgement signal is required. Business transaction protocol signals are independent from lower protocol and transport signals such as reliable messaging. The Business Signals are important for state alignment, and relate to the characteristics inherent in the BT patterns described earlier in Section 3. Business Signals and their relationship to success and failure are outlined in Section 3.6.3.

3.4.9.3.1 Receipt Acknowledgement Business Signal

The Receipt Acknowledgement Business Signal, if used, signals that a message (Request or Response) has been properly received by the BSI software component. The property isIntelligibleCheckRequired allows partners to agree that a Receipt Acknowledgement SHOULD confirm a message only if it is also legible. Legible means that it has passed structure/schema validity check.   If specified, the content of the receipt and the legibility of a business message (if required) MUST be reviewed prior to the processing of the Requesting or Responding Business Document or the evaluation of condition expressions in the message's Business Documents or Document Envelope. Condition Expressions are expressions that evaluate to true or false. Condition Expressions are described in more detail in Section 3.4.11. This property recognizes that the receipt and the legibility check may be handled separately with the latter completed prior to the Receipt Acknowledgement being generated. This attribute indicates the document is parsable and reusable. In addition, it may be advised to indicate that some industries, particularly that have EDI historical experience, may vary on 'syntactic check'. An implementation MAY also equate 'syntactic check' to using parser to validate the XML.

3.4.9.3.2 Acceptance Acknowledgement Business Signal

The Acceptance Acknowledgement Business Signal, if used, signals that the message received (Request or Response) has been accepted for business processing and that processing is complete and successful by the receiving application, service or a receiving business application proxy. This is the case if the contents of the business 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 document schema or Business 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 so that it 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 Acknowledgement. The substantive response would come after the Business Signal indicating whether the order had been Accepted or Rejected. Positive Business Signals or exceptions are non-substantive in nature, i.e. they may contain business identification data relevant to a business acceptance of an obligation (See definition of obligation earlier in Section 3). A substantive business message actually includes a Business Document such as a purchase order acceptance.

3.4.9.3.3 Business Signal Criteria

Based on any agreement between the parties, the requesting party typically MAY recognize that the Business Document had been successfully received and processed. Where applicable and used, the logical sequence of the Receipt Acknowledgement, Acceptance Acknowledgement and Response are based on the timing expectations defined. For example, in implementation, if an Acceptance Acknowledgement is received prior to a Receipt Acknowledgement, the requesting party may wait (if no timeout), for the Receipt Acknowledgement because the two Business Signals are handled by different systems. Occurrence of Business Signals and their receipt are not dependent. Occurrence is summarized in Section 3.5.1.

Business protocol engines are expected to deal with the precedence of the receipt of Business Signals. Many eBusiness systems are completely asynchronous, whereby there is no way to guarantee that physical receipt will be sequenced. Logical receipt however is sequenced.

Failure to send either signal, when required (by specifying a timeout value in timeToAcknowledgeReceipt or timeToAcknowledgeAcceptance), SHOULD result in the transaction being null and void. A control Failure has occurred. The transaction will not reach any "Success" end state. A "Success" end state (Protocol or Business) is dependent on receipt of a Business Document satisfying the associated TimeToPerform. In order for a BTA instance to reach a “Success” state at run-time, the following things SHOULD be true:

- no timeout would have occurred (signals or response)
- no signal can have a negative content
- the response document sent to the requester MUST be marked as isPositiveResponse = ‘true’ in the ebBP instance that specifies the Business Collaboration in order to support Business Success

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 transaction (although it is relevant to Business Success and Failure). 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 (Business) 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 BTA is considered successful, again after it has reached a Protocol Success state.

For example in the case of a Decision (linking construct), isPositiveResponse is in effect within a Decision related to the DocumentEnvelope. This is evidence of the preference to evidence collaborative shareable) information (i.e. the DocumentEnvelope) to align state between the parties involved.

Condition guards on transitions are discussed in detail in Section 3.6.3.

It is important to note that the isPositiveResponse attribute such as other facilities in ebBP - condition guards on transitions, semantic variables, conditions expressions - are enabling mechanisms for the ebBP process definitions whereby the choreography, control flow, state transitions, logical business documents, and the expectations of the parties are clearly understood. It is their collective use that provides the capability to enable Business Collaborations.

A corresponding isPositiveSignal attribute occurs on each signal. Although consistent with the structure of the Document Envelope, this attribute on each signal type has a fixed value.

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 the sending party that Business Documents, a request or a response have been successfully processed by the receiving application, whatever it might be. However, to achieve this result, the Business Transaction protocol MUST 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’s 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 it is known 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 MAY be relaxed and regular messaging services MAY be used.

Note, in order to guarantee the successful synchronization of state between two parties, reliable messaging MUST be used and the Business Transaction MUST be defined to use the request and response Acceptance Acknowledgement signals. When a Document Envelope exists, these signals are important to guarantee that the corresponding Business Documents were processed by the respective applications. Criteria surrounding the use of the Business Transaction patterns may include reliable messaging and use of the isGuaranteedMessageDelivery requirement (See Section 3.4.9.1). Any agreement between trading partners could specify that the certificate-based digest used by a message protocol could be captured and stored as the non-repudiation digest (making the message receipt function as a business protocol receipt). By default the Receipt Acknowledgement (and its associated on-repudiation attributes) are separate from the reliable messaging layer. In preceding technical specification versions, the guiding principles used were incomplete in describing the scope and operational details related to state synchronization. State synchronization may relate to the design and operational view of a business process specification like ebBP. In providing further concrete detail on the BT patterns, this technical specification concentrates on the operational view. Further business requirements may be identified from a design and modeling perspective that will affect these operationally focused patterns.

The difference between a Business Signal and a business message is that a signal has a fixed structure under the control of the infrastructure while a business message content may vary both at run-time and over time and is under the control of an application or service. ebBP technical specification specifies a schema for all signals of the Business Transaction protocol. However an extension mechanism is provided to support other schema definitions for Business Signals whereby user communities may define their own signal structure.

The Signal element is used to specify both ebBP and user defined signal schema references. The use of either is supported via the signal references in the ebBP and the Business Signal schema.

The logical relationship between the ebBP, Business Signal and underlying messaging are visible via the schema constructs. In addition to this technical specification and its appendices a non-normative package of ebBP and signal instances is available on the OASIS web site.

3.4.9.4 Sample syntax

Here is a slightly more complex transaction with two Document Flows and all 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.

```

<!--…-->
<Signal name="ReceiptAcknowledgement" nameID="ra2">
   <Specification location="http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0"
      name="ReceiptAcknowledgement" nameID="rabpss2"/>
</Signal>
   <Signal name="ReceiptAcknowledgementException" nameID="rae2">
      <Specification location="http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0"
         name="ReceiptAcknowledgementException" nameID="raebpss2"/>
   </Signal>
   <Signal name="AcceptanceAcknowledgement" nameID="aa2">
      <Specification location="http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0"
         name="AcceptanceAcknowledgement" nameID="aabpss2"/>
   </Signal>
   <Signal name="AcceptanceAcknowledgementException" nameID="aae2">
      <Specification location="http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0"
         name="AcceptanceAcknowledgementException" nameID="aaebpss2"/>
   </Signal>
   <Signal name="GeneralException" nameID="ge2">
      <Specification location="http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0"
         name="GeneralException" nameID="gebpss2"/>
   </Signal>
   <CommercialTransaction name="CreateOrder" nameID="ID110" isGuaranteedDeliveryRequired="true">
      <RequestingRole name="COinitiator" nameID="COinitiator1"/>
      <RespondingRole name="COresponder" nameID="COresponder1"/>
      <RequestingBusinessActivity name="sendOrder" nameID="ID111"
         isNonRepudiationReceiptRequired="false" isNonRepudiationRequired="false"
         timeToAcknowledgeAcceptance="PT1H" timeToAcknowledgeReceipt="PT1H">
         <DocumentEnvelope name="Purchase Order" nameID="ID112" businessDocumentRef="ID1010"/>
         <ReceiptAcknowledgement name="11011" nameID="ID11011" signalDefinitionRef=" ra2"/>
         <ReceiptAcknowledgementException name="11012" nameID="ID11012" signalDefinitionRef=" rae2"/>
         <AcceptanceAcknowledgement name="11013" nameID="ID11013" signalDefinitionRef="aa2"/>
         <AcceptanceAcknowledgementException name="11014" nameID="ID11014" signalDefinitionRef="aae2"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name="sendPOAcceptance" nameID="ID113"
         isNonRepudiationReceiptRequired="false" isNonRepudiationRequired="false"
timeToAcknowledgeReceipt="P1D">
         <DocumentEnvelope name="Reject Order" nameID="ID114" isPositiveResponse="false"
businessDocumentRef="ID1011"/>
         <DocumentEnvelope name="Accept Order" nameID="ID115" isPositiveResponse="true"
businessDocumentRef="ID1012"/>
         <ReceiptAcknowledgement name="11311" nameID="ID11311" signalDefinitionRef=" ra2"/>
         <ReceiptAcknowledgementException name="11312" nameID="ID11312" signalDefinitionRef=" rae2"/>
         <AcceptanceAcknowledgement name="11313" nameID="ID11313" signalDefinitionRef=" aa2"/>
         <AcceptanceAcknowledgementException name="11314" nameID="ID11314" signalDefinitionRef=" aae2"/>
      </RespondingBusinessActivity>
   </CommercialTransaction>
<!--…-->
```

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. Therefore, the

Receipt Acknowledgement for the PO Acceptance sent by the Requester will be received from the Requester by the Responder.

3.4.9.5 Business Document Flows

Request and Response Document Flows contain Business Documents that pertain to the Business Transaction request and response. These Business Documents have varying structures. 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 MUST always only one named Document Envelope for a Requesting Activity. There MAY be zero, one, or more 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. A Requesting and Responding Business Activity MUST exist for each Business Transaction (and associated Business Transaction pattern). This condition even applies to the Notification or Information Distribution where a Document Envelope and Business Document are not used. As indicated, the Responding Business Activity is important irrespective of a Document Envelope.

If multiple Document Envelopes occur in the Responding Activity, only one SHOULD be used. The condition expressions assist in specifying how a particular DocumentEnvelope may be identified and handled. Typically, different responses necessitate separate names that are identifiable by a NameID for reference.

In the actual execution of the purchase order transaction, however, only one of the defined possible responses SHOULD be sent and the others SHOULD NOT occur. In the case of PartialPurchaseOrderAcceptance, multiple partial responses may be handled separately via the choreography. Choreography is discussed in more detail in later in Section 3.

Each Document Envelope carries exactly one primary (logical) Business Document. That logical primary Business Document MAY map to more than one physical document. The constraint of one logical Business Document for one Document Envelope associated with a Requesting Business Activity does not restrict what happens in transmission. For example, many Business Documents may be sent together in a transmission envelope (and that each map to a logical Business Document in a Document Envelope).

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 unit of work in the payload in the ebXML Message Service message structure. Variables and condition expressions support identification of logical conversations between parties. Variables and condition expressions reference the content of the primary Business Document and not the content of the attachments. Condition Expressions and Variables are described in further detail later in Section 3.4.11.

Attachments are considered unstructured, such as an image. They are not interrogated within the Document Envelope, i.e. condition expressions and variables MUST not used on them. The Attachment construction has been made consistent with the logical Business Document. In addition, Attachments can be specified as optional. These changes have been added to meet provided user community requirements.

3.4.9.6 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="ID122A3F613C ">
      <Specification name="CreditRequestSchema" nameID="ID123A3F613D" type="schema"
          location="http://www.example.com/creditRequest.xsd"
targetNamespace="http://www.example.com/creditRequest"/>
  </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="ID122A3F8E3">
      <ConditionExpression expressionLanguage="XPath1" expression="//@CreditResponse=denied"/>
      <Specification name="CreditResponseSchema" nameID="ID123A3F613E" type="schema"
          location="http://www.example.com/creditResponse.xsd"
targetNamespace="http://www.example.com/creditResponse"/>
  </BusinessDocument>
  <BusinessDocument name="Credit Approved" nameID="ID122A3F6C3">
      <ConditionExpression expressionLanguage="XPath1" expression="//@CreditResponse=approved"/>
      <Specification name="CreditRequestSchema" nameID="ID123A3F613F" type="schema"
          location="http://www.example.com/creditResponse.xsd"
targetNamespace="http://www.example.com/creditResponse.xsd"/>
  </BusinessDocument>
  <BusinessDocument name="Credit Rating" nameID="ID122A3F8E4">
      <Specification name="CreditRatingSchema" nameID="ID123A3F613G" type="schema"
          location="http://www.example.com/creditRating.xsd"
targetNamespace="http://www.example.com/creditRating.xsd"/>
  </BusinessDocument>
  <CommercialTransaction name="Check Credit" nameID="ID122A3DD33" isGuaranteedDeliveryRequired="true">
      <RequestingRole name="CCinitiator" nameID="CCinitiator1"/>
      <RespondingRole name="CCresponder" nameID="CCresponder1"/>
      <RequestingBusinessActivity name="checkCredit" nameID="ID122A3E833"
          isAuthorizationRequired="true" isIntelligibleCheckRequired="true"
          isNonRepudiationReceiptRequired="true" isNonRepudiationRequired="true"
          timeToAcknowledgeAcceptance=" PT30S" timeToAcknowledgeReceipt=" PT10S">
          <DocumentEnvelope name="DE" nameID="IDDE1" isAuthenticated="persistent"
              isConfidential="persistent" isTamperDetectable="persistent" businessDocumentRef="ID122A3F613C"/>
          <ReceiptAcknowledgement name="122A3E834" nameID="ID122A3E834" signalDefinitionRef="ra2"/>
          <ReceiptAcknowledgementException name="122A3E835" nameID="ID122A3E835"
signalDefinitionRef="rae2"/>
          <AcceptanceAcknowledgement name="122A3E836" nameID="ID122A3E836" signalDefinitionRef="aa2"/>
          <AcceptanceAcknowledgementException name="122A3E837" nameID="ID122A3E837"
signalDefinitionRef="aae2"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name="confirmCredit" nameID="ID122A3E863"
          isAuthorizationRequired="true" isIntelligibleCheckRequired="true"
          isNonRepudiationReceiptRequired="true" isNonRepudiationRequired="true"
timeToAcknowledgeReceipt="PT10S">
          <DocumentEnvelope name="DE21" nameID="IDDE21" isPositiveResponse="false"
             isAuthenticated="persistent" isConfidential="persistent"
              isTamperDetectable="persistent" businessDocumentRef="ID122A3F8E3"/>
          <DocumentEnvelope name="DE22" nameID="IDDE22" isPositiveResponse="true"
              isAuthenticated="persistent" isConfidential="persistent"
              isTamperDetectable="persistent" businessDocumentRef="ID122A3F6C3">
              <Attachment name="Credit Report" nameID="IDAT1" mimeType="application/xml"
                  businessDocumentRef="ID122A3F8E4" isConfidential="none"
                  isTamperDetectable="none" isAuthenticated="none">
                  <Documentation>Credit report included with message.</Documentation>
                  <Specification name="CreditReportSpec" nameID="IDCRS"
location="http://www.example.com/HowToProcessCreditReport.xhtml"/>
              </Attachment>
          </DocumentEnvelope>
          <ReceiptAcknowledgement name="132A3E863" nameID="ID132A3E863" signalDefinitionRef="ra2"/>
          <ReceiptAcknowledgementException name="142A3E863" nameID="ID142A3E863"
signalDefinitionRef="rae2"/>
          <AcceptanceAcknowledgement name="152A3E863" nameID="ID152A3E863" signalDefinitionRef="aa2"/>
          <AcceptanceAcknowledgementException name="162A3E863" nameID="ID162A3E863"
signalDefinitionRef="aae2"/>
```

```
      </RespondingBusinessActivity>
  </CommercialTransaction>
<!--…-->
```

See Section 3.5.5 for a discussion on document security parameters.

3.4.9.7 Business Transaction Activity

A Business Transaction Activity is the performance of a Business Transaction within a collaboration. Business Transaction definitions can be associated to any number of BTA elements. This means that the same Business Transaction can be performed by multiple Business Transaction Activities in different collaborations, or by multiple Business Transaction Activities in the same 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 opposite roles, meaning that after a purchase order has been accepted, either party could cancel it (for a certain period of time) using the exact same Business Document interchange.

The BTA conveys additional semantics that configure the particular performance of the Business Transaction it references. The BTA binds each abstract business partner to a role, and to the generic role in the BT.

A Business Transaction Activity MAY specify that this particular document interchange “has legal intent” via the hasLegalIntent attribute. This attribute is optional and means that particular activity that could represents a statement or commitment between trading partners, and their shared intent. Referencing the eCommerce Patterns v1.0 [http://www.ebxml.org/specs/bpPATT.pdf], the digital signature cannot in and of itself infer intent. Given parameters outside of this specification, this constraint may be used as a substantive and enforceable precondition on the BTA. The mechanisms in the BSI that provide the capability to support this precondition are:

- reliability
- document security: confidential, tamper detectable and authenticated
- non-repudiation
- authorization
- predictability

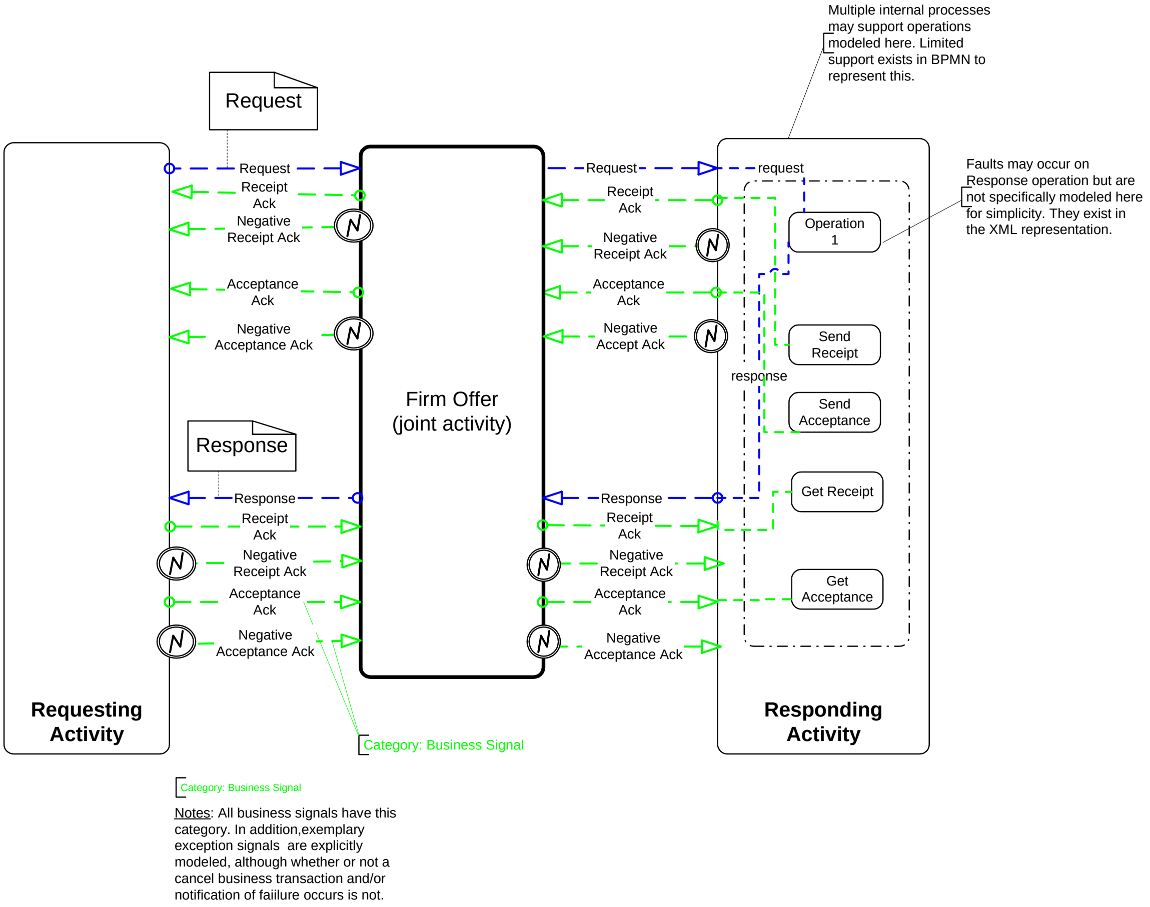
The parties may establish the parameters for reliability and intent, and its relationship to assurance or non-repudiation, for example. Agreements and enforceability may be relevant to establishing these capabilities. How these parameters translate to implementation decisions is unspecified. For example, it may be implemented using a receipt signature with digest, using and persisting digital signatures with ebMS, or other implementation options. Users may choose to use separate agreements to define business responsibility, including criteria for participation. The Requesting logical Business Document can trigger a chain of protocol-specified Responding documents and subsequent Business Transactions. Roles are bound to those Business Transactions.

The hasLegalIntent attribute could have widely differing interpretations and enforceability depending on type of business, process, and jurisdiction. No implication of interpretation or enforceability is made by the ebBP specification. The contractual framework, agreements and their application to any artifact are outside the scope of this specification. The implementer SHOULD NOT assume any particular runtime behavior based on this attribute.

3.4.9.8 Operation Mapping

An Operation Mapping specifies a possible mapping of a BTA to a set of web service operation invocations to enable the participation of a non-ebXML capable party in an ebXML relationship. An ebBP definition does not itself contain a reference to a WSDL file, but rather references to abstract operation names, which can be de-referenced with specific WSDL files, specified at the Collaboration Protocol Profile.

The goal of the Operation Mapping is to offer a flexible mapping scheme to map all Document Envelope and signal interchanges to any combination of web service operation interactions.



Multiple internal processes may support operations modeled here. Limited support exists in BPMN to represent this.

Faults may occur on Response operation but are not specifically modeled here for simplicity. They exist in the XML representation.

Notes: All business signals have this category. In addition,exemplary exception signals are explicitly modeled, although whether or not a cancel business transaction and/or notification of faiilure occurs is not.

Figure 8: A possible mapping between a Business Transaction definition and a set of operations

Note: Figure 8 was developed under the same assumptions as Figure 7 earlier in Section 3. A typical example is represented Figure 8. It shows that the Request and possible Responses of a Business Transaction Activity being mapped to a single operation invocation while the Business Signals are mapped to individual one ways and notifications or information (not the Notification Pattern). The mapping allows for any combination, where for instance a Request and a Receipt signal (one of the Business Signals) would map to a request/response operation. Similarly a Response document and an Receipt Acknowledgement signal could map to a solicit/response type of operation. There is no limit to the number of operations that can be mapped to a single BTA. In the context of BPMN v1.0, the Group object is used to show the mapping and relationship between the BTA and the associated possible abstract operations. The abstract operations are not subprocesses to the BTA but possible implementation choices for the activity.

The mapping is also designed to define an Operation Mapping on both sides of a BTA. This means that the ebBP specification can be used to define the abstract behavior of complex collaborations between web services even in the case where no role in the collaboration is capable of ebXML.

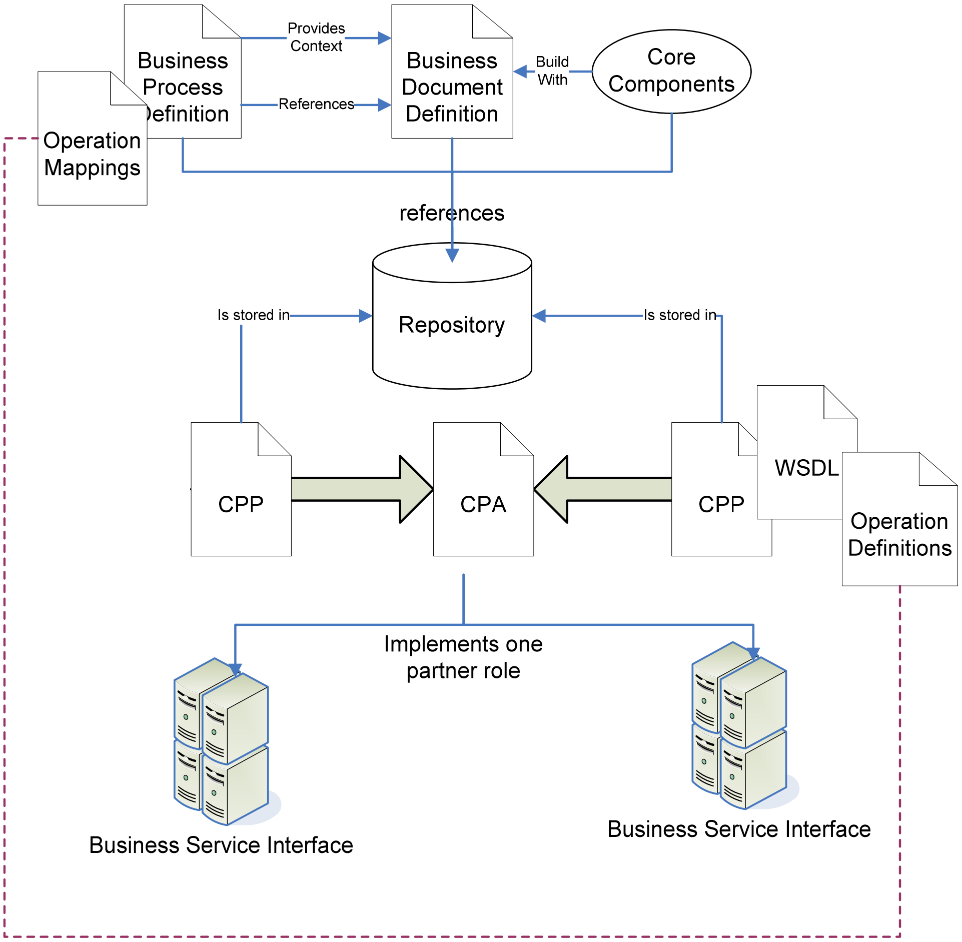


Figure 9: Operation Mapping in the ebXML architecture

Figure 9 presents the relationship of the Operation Mappings with the ebXML architecture. If a party is capable of web services only, it can create a simple Collaboration Protocol Profile which (1) references the WSDL files that contains the appropriate concrete operations and (2) may also include the service and actions that map to the ebBP process definition. More information is found in Section 3.5.7. The Business Collaboration definition processed by the BSI of the ebXML or correspondingly capable party will use the WSDL definition to identify or initiate the corresponding web service operation invocations.

The web service operations MAY be correlated to the corresponding ebBP instance and BTA. As of the time of this technical specification, a standards-based run-time correlation mechanism exists in the W3C (WS-Addressing) and WS-MessageDelivery offered as a W3C note. In the future, it is anticipated that this team will consider a white paper on how to use an addressing mechanism in the context of a BTA.

3.4.9.9 Sample syntax

This snippet example shows how a Catalog Request query response Business Transaction with a Supplier abstract partner role may be implemented with web services. If this example BTA was expanded, the Operation Mapping may include business messages and signals of that BTA including inputs, outputs and faults. The BTA defines the business messages and/or signals that MAY map to abstract operations. When the OperationMapping constructs are used, the abstract operations MUST map to the specified business messages and signals in the corresponding BTA (for full coverage of the BTA constructs).

```
<!--…-->
<QueryResponse name="Catalog Request" nameID="ID100" isGuaranteedDeliveryRequired="false">
    <RequestingRole name="QRinitiator" nameID="QRinitiator1"/>
    <RespondingRole name="QRresponder" nameID="QRresponder1"/>
    <RequestingBusinessActivity name="requestCatalog" nameID="ID101">
      <DocumentEnvelope name="Catalog Request" nameID="ID102" businessDocumentRef="ID1000"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name="sendCatalog" nameID="ID103">
      <DocumentEnvelope name="Catalog Response" nameID="ID104" isPositiveResponse="true"
        businessDocumentRef="ID1001"/>
    </RespondingBusinessActivity>
  </QueryResponse>
   <BusinessCollaboration name="BC" nameID="BC100">
    <Role name="Buyer" nameID="ID7902847"/>
    <Role name="Supplier" nameID="ID7902028"/>
    <TimeToPerform duration="P1D" type="design"></TimeToPerform>
    <BusinessTransactionActivity name="Catalog Request" nameID="ID100300"
      businessTransactionRef="ID100" hasLegalIntent="false">
      <TimeToPerform duration="P1D"/>
      <Performs currentRoleRef="ID7902847" performsRoleRef="QRinitiator1"/>
      <Performs currentRoleRef="ID7902028" performsRoleRef="QRresponder1"/>
    </BusinessTransactionActivity>
   <!—start and completion omitted-->
  </BusinessCollaboration>
   <OperationMapping name="Catalog Request" nameID="ID23948092" roleRef="ID7902028"
    businessTransactionActivityRef="ID100300">
    <MessageMap interfaceName="Procurement" operationName="catalogRequest" operationStep="input"
      documentEnvelopeRef="ID102"/>
    <MessageMap interfaceName="Procurement" operationName="catalogRequest"
      operationStep="output" documentEnvelopeRef="ID104"/>
   <!—fault omitted-->
  </OperationMapping>
<!--…-->
```

Note: In the preceding example, in a BTA context, Performs' currentRole attribute contains a value referring a Role by Requesting or Responding Role attributes that contain a value referencing a Requesting or Responding Business Activity and that relate to those identified in the Business Collaboration.

A more complex OperationMapping could be specified where roles change in BTA within a Business Collaboration and where different operations come from different interfaces.

3.4.10 Specify a Business Collaboration

3.4.10.1 Key Semantics of a Business Collaboration

There is no conceptual difference between a Binary and a Multiparty (Business) Collaboration. A Binary (Business) Collaboration is a Multiparty Collaboration between two roles only. However, architecturally, there is a difference. A Binary (Business) Collaboration is always self-coordinated, while a Multiparty (Business) Collaboration may require infrastructure level coordination to align the state of all relevant parties after any given message interchange. This type of infrastructure coordination is out of scope for the current version of the technical specification and it is assumed that Multiparty (Business) Collaborations will be designed with explicit Business Transactions to synchronize the state of the collaboration for the relevant parties. The BinaryCollaboration and MultipartyCollaboration elements are here for backward compatibility. Moving forward, collaboration definitions SHOULD be using the BusinessCollaboration element.

The context of a Business Collaboration is limited to the Document Flows, activities and signals 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.

One of the roles is initiating the Business Collaboration. This is the role (or may be associated with the role that performs the activity) that sends the first message (i.e. Request) of the first BTA. The initial abstract partner roles of the parent Business Collaboration are bound to the roles of an inner Collaboration Activity, when there is an inner Collaboration Activity. The abstract partner roles, the roles bound and performed (such as the currentRoleRef and performsRoleRef in the Performs element), and how they relate are described in detail in Section 3.4.1.

A Business Collaboration consists of one or more Business Activities. These Business Activities are always conducted between the two roles chosen from the roles of the Business Collaboration. For each activity one of two roles is assigned to be the initiating roles (from) and the other to be the Responding roles (to). This is irrespective of who actually initiated the Business Collaboration.

A Business Activity MAY either be a BTA, a Complex BTA or a Collaboration Activity.

A BTA is described earlier in Section 3.4.9.7.

A Complex Business Transaction Activity (ComplexBTA) allows for nested BTAs to happen in a recursive manner. This concept is a pure sequencing concept and does not affect the atomicity of the Business Transaction. The choreography mechanisms for the Business Collaboration allow for Business Transaction Activities to happen in parallel, however there MAY be a need to express that a BTA can happen only after the request of the other BTA has been entirely processed (including the return of acknowledgements). This is precisely the purpose of Complex Business Transaction Activity. When multiple activities are nested within ComplexBTA, these activities MUST be executed in series. The model supports for any number of nesting levels. Each activity element is associated with a StatusVisibility element that specifies which state (Success, Failure and document exchanged) are visible at the level of the parent ComplexBTA.

The ComplexBTA provides a mechanism to implement and communicate the dependencies between an actual business process (semantic process) and systems implementation of business processes (service choreography). An actual business process may subscribe to events happening in the services layer, and update the actual state when the event is received. This functionality allows a complete decoupling of the implementation, as well as clear view of the required information at the actual (real world) business layer. This mechanism allows the status to be known and published in a Business Collaboration with the default being no status visibility. When status visibility is desired for a ComplexBTA, a simple scenario is provided: Assume a Buyer and Seller are parties to the Business Collaboration. The Seller may have visibility to other sub-parties, such as Suppliers, and is responsible for the performance of the sub-parties. In this sense, the sub-parties are not first class citizens to this particular Business Collaboration nor constrained by it. Another Business Collaboration may exist elsewhere that defines the interaction of the parties that are sub-parties visible in this Business Collaboration. Conversely, in a Multiparty (Business) Collaboration, the parties are responsible in that Business Collaboration. For example, the Supplier would be responsible for the performance of the sub-parties. A brief example of a ComplexBTA is shown in Figure 10.

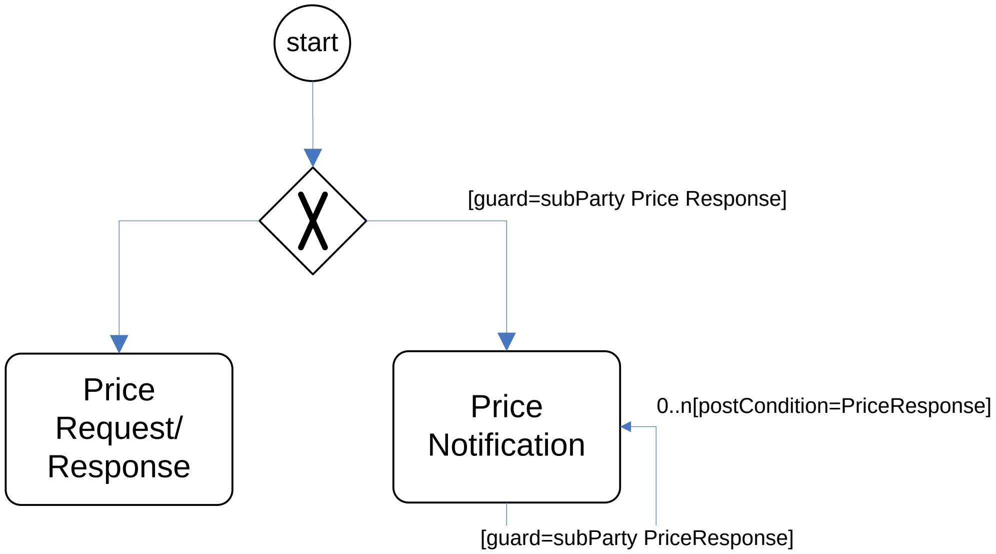

start
[guard=subParty Price Response]
Price Request/ Response
Price Notification
0..n[postCondition=PriceResponse]
[guard=subParty PriceResponse]

Figure 10: Status Visibility

For ComplexBTA, the Status Visibility is included in order to specify which status values of the embedded processes are considered, if any, when returning the status value to the context in which the parent ComplexBTA was included.

Condition expressions and guards govern the incoming transitions on links (FromLink from a parent ComplexBTA for example). Each of the FromLinks can be specified to transition to the CompletionState (Success or Failure) as a result of the satisfying condition guard. This allows, for example, exposing technical failures. If expected, failures can also be modeled. The parties specify how it is handled. Condition expressions and variables are described in Section 3.4.11. Expected (choreographed) and unplanned (General Exceptions) are described further in Section 3.6.2.3.

As described later in Section 3, these linking constructs, or movements between states (which were previously called pseudo-states), would be Start, CompletionState (and sub-specializations of that, Success and Failure), Fork, Join, Decision (or Choice), and Transition. They correspond to bundles of labeled edges of a directed possibly cyclic graph. At their core, they are collections of pairs of nodes, and describe the potential paths of a ebBP definition.

In the ComplexBTA, this nesting and the associated constraints allow monitoring of the state model of the collaboration and specifies event visibility of the service layer model. The ebBP state model and expression enumerate semantic business events and the complexities of service events are mapped at a technical layer onto business events (semantic business occurrences). This decoupling is extremely powerful as incremental improvements in both service and business layer evolve. If a business process designer specifies the Document Flow from another sub-party, it becomes visible. This allows incremental progress in order to anticipate and accommodate future development needs by enabling status visibility in a nested process. Other capabilities evolving in the messaging layer (such as in future versions of ebXML Messaging Service) may also support this projected status requirement.

Such capabilities allow more effective monitoring of the activities defined. The process designer may choose to use the status visibility details as input to make decisions on other business logic used in this enclosing BTA. Industry sectors such as logistics processes (particularly for international trade) may make use of this mechanism to allow migration to global, potentially fully visible, collaborations between many parties.

The nesting for status visibility and transitions in a ComplexBTA is unbounded. More business requirements are being gathered to determine the need and use of status visibility in other activities such a Business Collaboration (Multiparty) and the utility of administrative monitoring. In the future, it is also anticipated that managing coordinated, complex activities and visibility will be expanded for Business Collaboration of more than two abstract partner roles and for ComplexBTA. Such coordination may expand the relationship of the ebBP technical specification to other emerging specifications and technologies, in order to support specialized status visibility, particularly to further enhance monitoring capabilities.

A Collaboration Activity is the performance of a Business Collaboration, within another Business Collaboration. Business Collaboration definitions are re-useable relative to Collaboration Activity. The same Business Collaboration can be performed by multiple Collaboration Activities in different collaborations, or by multiple Collaboration Activities in the same Binary (Business) Collaboration. A Binary (Business) Collaboration definition may be restricted to be an “inner collaboration” only via the boolean attribute isInnerCollaboration. In this case, the Binary (Business) Collaboration definition can only be initiated as part of a Collaboration Activity and cannot be initiated by itself. The isInnerCollaboration attribute MAY occur on any Business Collaboration and specify it MAY only occur from within another Business Collaboration.

Business Transaction Activities, Complex Business Transaction Activities and Collaboration Activities MAY define business rules with the BeginsWhen, EndsWhen, PreCondition and PostCondition elements. These elements MAY be used for annotation purposes. If the expressions rendered as computable, the BSI MAY use them at run-time.

These element definitions are:

- PreCondition: A description of a state external to this activity that is required before the activity can commence.
- PostCondition: A description of a state external to this activity that is required after the activity concludes (i.e. the state doesn't exist before the execution of this activity but does exist afterwards).
- BeginsWhen: A description of an event external to this activity that normally causes it to commence (i.e. PreCondition + other variables = BeginsWhen).
- EndsWhen: A description of an event external to this activity that normally causes it to conclude (i.e. PostCondition + other variables = EndsWhen).

These expressions may also be made available elsewhere (such as used internally) to further verify the legitimacy of an activity. The partners involved collaboratively define the constraints whereby they engage in these activities. This may provide the capability for both parties to verify the conditions (rules or logic, for example) were followed.

If desired, variables MAY be used to further enable Pre- and PostCondition, BeginsWhen and EndsWhen elements, as they are of type ConditionExpressionType. For example, an XSLT variable may be used for the expression of this condition and allow values to be placed in them. Variables are semantic enablers, as discussed in Section 3.4.11.

It is possible that conditions, such as these, could be a part of a standard application of a Business Transaction and/or specific to the context of which the transaction that is used (for a Business Transaction Activity). If conditions existed on the BT, they could act as process gatekeepers into/out of the BT. Enabling conditions on the BT (in addition to where they currently exist on the BTA) may be considered in a future version.

The semantics of BeginsWhen and EndsWhen indicate that the corresponding Business Activity is expected to be started or ended as soon as the expression in the attribute value is true. The BeginsWhen expression MAY be used to:

- Link a semantic state (e.g. begins when "state" of "product-delivered" is reached)
- Serve as a semantic definition that MAY be used to define that state (e.g. "in the context of this ebBP definition, "product-delivered" is defined as the existence of both product-delivered date and delivery-signature)

These external events may drive a transition and condition to be possible or not (and hence could affect success or failure). For example, an invoice may not be generated until a product is delivered.

For EndsWhen, in the case of a certification exam, a registrant is allowed three attempts to pass an exam to achieve certification; otherwise the registrant fails. In an academic setting, a health care provider, i.e. the registrant, attempts the certification exam three times. For the first try, the registrant submits a certification request and engages in a registration step. The registrant request fails and is returned. The registrant increases insurance, retries and fails. For a third try, the registrant increases staff capacity, then retries. The registrant requests fails a third time. The registrant attempts to re-register but must start over again. This scenario may apply to other than health care, such as Amazon self-registration.

The EndsWhen is a quality of service attribute that may enable evaluation (and in the future computation) of Business Transaction status after the Business Document is received. EndsWhen may be a description of an event external to this collaboration that typically causes this collaboration to conclude.

A PreCondition indicates that the corresponding Business Activity may start only if the corresponding expressions are true. A PostCondition expresses a condition that must be true once the activity has been completed. For example, Business Success is true (i.e. the status reported to the choreography is true) when the activity is completed.

Whether BeginsWhen, EndsWhen, or Pre- or PostCondition, the information MUST be visible to the parties involved.

In the future, these capabilities could be filter- or subscription-based capabilities to enable the business community to define the semantic business-event controlling the process. A constraint may be declared on an action that maps to information that is produced by that action. For example, BeginsWhen is based on business content in the business message delivered on that action.

Such constructs may be useful for process-context driven communication, monitoring and verification of rules related to content driven processes. For example, a Business Collaboration requires a notification of delivery. A DeliveryNotification transaction adheres to the Notification pattern is used that includes a Receipt Acknowledgement signal. However, the parties involved only want that notification to take place when the signature is available. This could occur when the driver return his device, although implementation (result) is visible to the business process. The transition occurs to this transaction as soon as the product is shipped, so the enabling component is then, in essence, waiting for an event that will start this transaction.

When performing a Collaboration Activity within a collaboration there is an implicit relationship between the roles at multiple levels (two at a minimum). For example, assume that a Binary (Business) Collaboration Firm Order is performing Binary (Business) Collaboration Product Fulfillment through Collaboration Activity Drop Ship. Binary (Business) Collaboration Firm Order has the following roles: Customer and Retailer. In Collaboration Activity Drop Ship we assign Customer to be the Initiator, and Retailer to be the Responder. Binary (Business) Collaboration Product Fulfillment has the following roles: Buyer and Seller and a BTA where Buyer is the Initiator and Seller is the Responder. The Business Transaction and its declared roles are used by the BTA. We have now established a role mapping and relationships between the roles

Customer and Buyer because they are both Initiators in activities in the related performing and performed Binary (Business) Collaborations.

Since a Business Transaction is atomic in nature, the performing of a single Business Transaction through a BTA is also atomic in nature. If the desired semantic is not atomic, and then the task SHOULD be split over multiple Business 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 (Business) Collaboration and the partial acceptance(s) as separate transactions, thus handling the partial acceptances within the choreography. The choreography can also support multiple requests in the same manner.

Similar or more complex circumstances may apply. For example, the Document Envelope may contain multiple EDI (Electronic Date Interchange) payloads or pertain to separate Business Transactions. In this case, it is recommended that choreography be used to logically handle these, similar to how multiple requests or responses are handled. More requirements will be solicited to evaluate what other mechanisms are needed to support Business Collaboration and conditions such as those that may apply to batch processing.

The parties can agree upon a CPA in order to transact business. A CPA may associate itself with a specific collaboration. Thus, all Business Transactions performed between any two parties SHOULD be referenced through Business Transaction Activities contained within a Business Collaboration.

For a Business Collaboration involving more than two parties, the roles assumed by the parties MUST be specified. The Performs element MUST be used to assign the roles that a party assumes for this type of Business Collaboration. Where allowed, the Performs element MAY be omitted if the actual values of Roles in the referring and referred-to context are the same (i.e. string identical) and they match. If a new value is found in the referred-to context and it has not been associated with a previous role, then it MUST be considered to be a new role.

A party may assume several roles during a Collaboration Activity. When a Business Collaboration between two parties is related to another Business Collaboration (also of two parties) using a Collaboration Activity, the roles may change for the parties. Those roles MUST be traced and associated with the parties. For example, a Handle Order Business Collaboration (of two parties) invokes a CreditCheck via a Collaboration Activity. The Seller (in the top level Business Collaboration) also performs the role of Customer and the Credit Agency also performs the role of Credit Service.

This functionality supports tracing and binding of roles of the Business Collaboration across and within multiple levels of nesting. Roles can be mapped and referenced (via @nameID) through multiple levels of activity nesting.

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 BTA can be ‘open’ at the same time within any Business Collaboration instance performed between any parties. isConcurrent limits the ability to execute multiple BTA of the same BT across Business Collaboration instances (with the same party), or within the same Business Collaboration if multiple paths are open.

As a result, when isConcurrent is set to false, the BSIs of each party MUST serialize these BTAs.

3.4.10.2 Sample syntax

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

```
<BusinessCollaboration name="Firm Order" nameID="ID122A38D93">
   <Role name="buyer" nameID="ID122A38DA3"/>
   <Role name="seller" nameID="ID122A38DA5"/>
   <TimeToPerform duration="P1D"/>
   <Start name="ID876F38OP5" nameID="ID876F38OP5">
      <ToLink toBusinessStateRef=" IDPO3DA1"/>
   </Start>
   <BusinessTransactionActivity name="Place Order" nameID="IDPO3DA1"
      businessTransactionRef="ID122A3DD33" hasLegalIntent="true">
      <TimeToPerform duration="PT4H"/>
      <Performs currentRoleRef="ID122A38DA3" performsRoleRef="CCinitiator1"/>
      <Performs currentRoleRef="ID122A38DA5" performsRoleRef="CCresponder1"/>
   </BusinessTransactionActivity>
   <Success name="Success" nameID="D2JSK99AK"/>
   <Failure name="Failure" nameID="DK9726AJ"/>
      <Decision>
      <FromLink fromBusinessStateRef=" IDPO3DA1"/>
      <ToLink toBusinessStateRef=" D2JSK99AK">
         <ConditionExpression expressionLanguage="ConditionGuardValue" expression="Success"/>
      </ToLink>
      <ToLink toBusinessStateRef="DK9726AJ">
         <ConditionExpression expressionLanguage="ConditionGuardValue" expression="Failure"/>
      </ToLink>
   </Decision>
</BusinessCollaboration>
```

3.4.11 Choreography

3.4.11.1 Key Semantics of a Choreography

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

The choreography is specified in terms of Business States, and transitions between those Business States. 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 (Business) Collaboration. It is merely the execution of the backend systems, which instruct the BSI to send or receive messages that advance the state of a collaboration. There is no execution engine associated to the collaboration itself.

The Business Collaboration is either in the state of performing a given Business Activity (or multiple concurrent Business Activities) or waiting to start a Business Activity, unless it has reached a completion state. Once a Business Activity completes a transition from this Business Activity, it navigates to another Business Activity. A business message initiates a Business Collaboration or advances its state.

There are a number of auxiliary kinds of States that facilitate the choreographing of Business Activities. These include a Start state, a Completion state (which comes in a Success and Failure flavor) as well as a series of gateways: a Fork gateway, a Join gateway and a Decision gateway. There are two types of Fork gateway: OR and XOR.

An XOR Fork means that only one Business State of the Fork will be allowed to be reached, although all transitions to Business States are possible at the start. Once one of the outgoing transitions attached to the Fork gateway get activated, all the other transitions becomes invalid (e.g. a BTA starts).

An OR value mean that one or more Business Activity pointed to by a transition coming from the Fork might be initiated. Several paths are possible although when and which become active is unknown. 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 arrive that determines the order in which the Condition Expression need to be evaluated. A Decision differs from a Fork in the sense that a Decision selects only one of the possible transitions, and the other(s) is/are automatically disabled. An XOR Fork may be designed to operate like a Decision, but a Decision cannot be an XOR Fork.

A Fork has a TimeToPerform element, where the duration MAY be specified. At the end of this time interval, the state of the Binary (Business) Collaboration will automatically be moved to its corresponding Join. This feature MAY be used in cases where the Business Activities are optional. For instance a Cancel Purchase Order and Change Purchase Order BTA could be defined as part of a Fork/Join control block. However, most often none of these activity would happen. If any given BTA within the Fork/Join pair has not reached its completion state, the BSI will generate a corresponding timeout exception. The TimeToPerform duration of a Fork MUST be greater than (or equal to) any TimeToPerform duration of its Business Activities.

Via the AND-Join (by default, the Join is an AND-Join), the waitForAll attribute (waitForAll='true') of the Join MUST indicate that all transitions coming into the Join MUST be executed for the collaboration to reach the Join state that reflects the state movement. When the waitForAll parameter is set to false, it is an OR-Join. If one or more Business Activities complete, the OR-Join (waitForAll='false') completes. For an OR-Join (where waitForAll='false'), 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 element MUST NOT be used. It MUST only be used in the case where all outgoing transitions from the Fork have incoming transitions to the Join.

For XOR or OR Fork, this does not rule out different joins pertaining to states emerging from a Fork or Forks. This allows a split in processing between a group all of which must be done and one where at least one (or more) is sufficient for the transition. As bounded by Fork semantics, multiple joins may be allowed for a fork (multiple dependencies exist). The behavior of Forks over Joins may be handled by monitoring capabilities (for example, detection via static analysis).

Fork	Join	Comments
OR	waitforAll (true)	This models the behavior of an AND-Fork and AND-Join
OR	waitforAll (false)	The Join state is reached when the activity has been performed or when the timeout occurs, whichever comes first. TimeToPerform on a Fork is typically used when a Join is expected to be taken (i.e. the Join takes place even if the activities do not).
XOR	waitforAll (true)	This combination is forbidden (creates a dead lock)
XOR	waitforAll (false)	Only one path between the Fork and Join will be allowed to happen
TimeToPerform Duration >0	Any value	The Join happens when TimeToPerform duration is reached.

Table 7 TimeToPerform

Forks and joins are useful particularly when activities between parties may be optional. For example, in retail or manufacturing/production cases, order status may or may not occur. However, when it does occur, the order response and status are important to the involved parties. In another case between a Producer and a subcontractor, the order status, a disposition change and response, and other integration changes may or may not occur. In both cases, these optional business transactions may be modeled as forks between the related business transactions.

Transitions can originate from Business Transaction Activities, Complex Transaction Activities or Collaboration Activities within a Business Collaboration. Guards MAY gate transitions. Guards refer to the status of the Activity from which the transition originates. The guard values include: ProtocolSuccess, AnyProtocolFailure, RequestReceiptFailure, RequestAcceptanceFailure, ResponseReceiptFailure, ResponseAcceptanceFailure, SignalTimeOut, ResponseTimeOut, Failure, BusinessSuccess, BusinessFailure and Success.

3.4.11.1.1 Use of Variables and Condition Expressions

Transitions MAY also have a Condition Expression element. Condition expression MAY depend on variables. Variables are named information elements that are available to bind concepts across Business Transaction. They also serve to make the semantics clear in a condition expression. There are two types of variables: simple and complex. Simple variable reference a BTA and a Business Document exchanged as part of this BTA. Variables allow abstract elements used in conditional statements as well as external specifications (e.g. business agreements) to link to Business Document contents. For example, variables may be used to apply context to a particular business transaction and the roles involved. The capability to bind semantic information raises visibility to what drives the execution of the Business Collaboration.

Each Variable represents an abstract information element, and is defined by XPath executed on a Business Document instance. Once defined a variable MAY be used in any conditional statement as a node-list in the condition XPath. For instance if two variables are defined:

```
<Variable  name="PO Accepted" nameID="H7YIUSOP" businessTransactionActivityRef="ID122A39C23"
businessDocumentRef="ID1012">
    <ConditionExpression expressionLanguage="XPath1" expression="//POAck[@status=’Reject’]"/>
</Variable>
```

The implementation of the collaboration engine MAY compute these variables whenever a document they are defined on is processed. Each occurrence of the variable would be maintained, and the entire list of occurrences of each passed as a node list to any component evaluating a condition statement.

The lists may be kept in order, so that multiple lists can be indexed to each other. For instance, in a negotiation, if $quoteAmount[1] is referenced the first quote amount is acquired.

The Variable element allows a Business Document instance to be referenceable. For instance $order.request would contain a reference to the Business Document instance for the business message (“request”) for the businessTransactionActivity (“order”).

These variables could be made externally available for use, such as for a business agreement. Control of multiple instances will be handled in implementation.

Typically simple variables are implemented with the XPath language and extract values from a given Business Document. If a BTA is executed multiple times, an array of values is automatically created for this variable. Complex variables contain complex expressions, which can reference other variables. A simple variable cannot reference another variable. Complex variables are typically specified with XSLT, which enables the passing of variables as an input to the XSLT execution. A ConditionExpression element MAY be associated to a variable, which can be either Boolean or Decimal. When the variable is of decimal type, it is casted as “true” when it is greater than zero and to “false” otherwise. Alternatively a ConditionExpression also has an optional language attribute, which specifies in which language the predicate is written. One such expression language is a DocumentEnvelope (expressionLanguage of ExpressionLanguageType), which allows specifying the exchange of a particular response document type, by the Business Transaction Activity from which the transition initiates.

This specification does not limit the type and number of languages a BSI MAY support for variables or condition expressions. A BSI MUST support at least two forms of the ConditionExpression element: the XPath language, as well as the DocumentEnvelope (of ExpressionLanguageType). This ExpressionLanguageType is simply defined as the nameID of a DocumentEnvelope. This expression language type was known in preceding ebXML BPSS versions 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 (Business) Collaboration instance. XPath may also operate on the result of rendering EDI into XML per ISO/DIS20625. When the DocumentEnvelope of ExpressionLanguageType is used for an expression, the nameID of the DocumentEnvelope SHOULD be used. More details on the use of NameID for referencing is found in Section 3.8.

In addition, other functions have been identified where variables may be used. Variables MAY provide the capability to redefine timing expectations during the product lifecycle. The use of variables in this way is described later in Section 3.

XPath SHOULD be and XSLT (Extensible Stylesheet Language Transformation) MAY be used, particularly when multiple condition expressions and variables are used. Currently or in the future, other technologies may also support the use of condition expressions and variables include XQuery (W3C), OASIS CAM or others.

The Success and Failure elements represent completion states. The FromLink element ensures that a transition to a completion state MAY be guarded by a conditionGuard. The Success or Failure of the collaboration does not affect the Success or Failure of the individual BTAs, which comprise the Business 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 MAY 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 open Activities. The BSI MUST reject all further messages associated to a collaboration instance as soon as a completion state is reached.

In this version, the condition expression and variable functions allow assignment of the TimeToPerform value through the process lifecycle to enable late binding. The TimeToPerform element MAY specify a duration and a type (for example, the value MAY be specified at design time). More requirements will be gathered to further understand the definition, use and other scenarios where variables may apply.

3.4.11.2 Sample syntax

Here is the same Binary (Business) Collaboration as used before, with choreography. There is a transition between the two, a start and two possible outcomes of this collaboration, Success and Failure:

```
<BusinessCollaboration    name="Firm Order" nameID="ID122A38D93">
   <Role name="buyer" nameID="ID122A38DA3"/>
   <Role name="seller" nameID="ID122A38DA5"/>
   <Role name="creditauthority" nameID="ID122A38DA7"/>
   <TimeToPerform duration="P1D"/>
   <Start name="ID876F38OP5" nameID="ID876F38OP5">
      <ToLink toBusinessStateRef="ID122A39C23"/>
   </Start>
   <BusinessTransactionActivity name="Place Order" nameID="ID122A39C23"
      businessTransactionRef="ID110" hasLegalIntent="true">
      <TimeToPerform duration="PT4H"/>
      <Performs currentRoleRef="ID122A38DA3" performsRoleRef="ID122A3E833"/>
      <Performs currentRoleRef="ID122A38DA5" performsRoleRef="ID122A3E863"/>
   </BusinessTransactionActivity>
<BusinessTransactionActivity name="Check Credit" nameID="ID122A39D24"
      businessTransactionRef=" ID122A3DD33" hasLegalIntent="true">
      <TimeToPerform duration="PT4H"/>
      <Performs currentRoleRef="ID122A38DA5" performsRoleRef="CCinitiator1"/>
      <Performs currentRoleRef="ID122A38DA7" performsRoleRef="CCresponder1"/>
   </BusinessTransactionActivity>
   <Success name="Success" nameID="D2JSK99AK"/>
   <Failure name="Failure" nameID="DK9726AJ"/>
   <Decision>
      <FromLink fromBusinessStateRef="ID122A39C23"/>
      <ToLink toBusinessStateRef="ID122A39D24">
         <ConditionExpression expressionLanguage="ConditionGuardValue" expression="Success"/>
      </ToLink>
      <ToLink toBusinessStateRef="DK9726AJ">
         <ConditionExpression expressionLanguage="ConditionGuardValue" expression="Failure"/>
      </ToLink>
   </Decision>
```

```
  <Decision>
    <FromLink fromBusinessStateRef="ID122A39D24"/>
    <ToLink toBusinessStateRef="D2JSK99AK">
      <ConditionExpression expressionLanguage="ConditionGuardValue" expression="Success"/>
    </ToLink>
    <ToLink toBusinessStateRef="DK9726AJ">
      <ConditionExpression expressionLanguage="ConditionGuardValue" expression="Failure"/>
    </ToLink>
  </Decision>
</BusinessCollaboration>
```

The completion states of this Business Collaboration definition are mutually exclusive.

Optionally the transition with the ConditionExpression could be expressed using variables based on an XPath predicate:

```
<Variable  name="PO Accepted" nameID="H7YIUSOP" businessTransactionActivityRef="ID122A39C23"
businessDocumentRef="ID1012">
   <ConditionExpression expressionLanguage="XPath1" expression="//POAck[@status=’Reject’]"/>
</Variable>
…

<Decision   name="Decision10" nameID="IDDecision10">
      <FromLink fromBusinessStateRef="ID122A39C23"/>
      <ToLink toBusinessStateRef="ID122A39D24" >
      <ConditionExpression expressionLanguage="XPath1” expression="PO Accepted" />
      </ToLink>
      <ToLink toBusinessStateRef="DK9726AJ" >
        <ConditionExpression expressionLanguage="ConditionGuardValue" expression="Failure"/>
      </ToLink>
   </Decision>
```

3.5 Core Business Transaction Semantics

The ebXML concept of a Business Transaction and the semantics behind it are central to predictable, enforceable commerce. It is expected that any 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 BSI software components to predictably and deterministically implement ebXML Business Transactions, allows you to specify electronic commerce transactions that provide

- Interaction Predictability, i.e. have clear roles, precise transaction scope, understood time bounds, succinct business information semantics, and unambiguous determination of Success or Failure. Each party can compute without ambiguity and the status of a transaction independently.
- Ability to show shared intent related to defined expectations between parties, i.e. the ability to specify that Business Transactions MAY be agreed to show intent of the parties.
- Non-repudiation, 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 characteristics are related to the BT patterns and supporting matrices referenced earlier in Section 3.4.9.1.

These available 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 Business Transactions. In particular, no claim is made that the ebXML Business Transaction concept covers all possible Business Transactions. For instance, a use case could involve exchanges of a request and two responses as a unit of work. The primary way to handle such a use case would be to specify in the choreography as a Binary (Business) Collaboration involving as many ebXML Business Transaction as necessary. The Binary (Business) Collaboration definition would then be specified in such a way to handle the individual ebXML Business Transaction exceptions and aggregate them. Therefore, the multiple responses are handled in the choreography itself.

3.5.1 Interaction Predictability

All Business Transactions follow a precisely prescribed flow, or a precisely defined subset thereof. 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 across company boundaries, we must use such protocol to achieve the business state synchronization as recorded by each party enterprise systems.

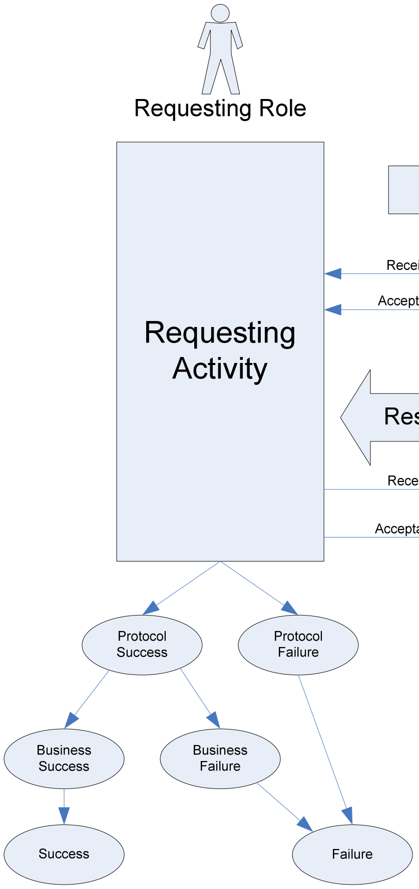


Figure 11: Schematic of core Business Transaction semantics

Figure 11 does not assume any hierarchy in the way exceptions are generated or evaluated. In order to achieve a Success state, a BTA MUST complete with both a Protocol and a Business Success. Exceptions are constantly evaluated by the BSI, and thrown as soon as detected. This is graphically represented in Figure 12 later in Section 3.

If either a Protocol or Business Failure occurs, the BTA will be put into a Failure state.

Only if agreed by the parties, a Notification of Failure MAY be issued during the performance of a Business Collaboration. At this point all further message exchange relative to this Business Collaboration instance is prohibited and will end in Failure.

In the ebXML model the Business Transaction has the following semantics:

- The Business Transaction is an atomic unit of work. All of the interactions in a Business Transaction MUST succeed or each party MUST revert their state to the state prior to the start of the BTA.
- A Business Transaction is conducted between two business partners playing opposite roles in the transaction. These roles are always the Requesting and Responding roles.
- 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.
- A Business Transaction always starts with a request sent out by the Requesting Activity.
- The Request serves to transition control to the Responding role.
- After the receipt of the Request Document Flow, the Responding Activity MAY send a ReceiptAcknowledgement Business Signal and/or an AcceptanceAcknowledgement Business Signal to the Requesting role.
- The Responding role then enters a Responding Activity. During or upon completion of the Responding Activity zero or one Response is sent.
- Control will be returned back to the Requesting Activity if either a ReceiptAcknowledgement and/or AcceptanceAcknowledgement and/or a Response are 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. Occurrence of Business Signals and their receipt are not dependent. Receipt is summarized in Section 3.4.9.3.3.

All Business Transactions succeed or fail. Success or Failure depends on:

- The successful transmission of the request, the response and/or receipt and acceptance signals
- The occurrence of time-outs
- The occurrence of exceptions, as indicated by a negative receipt or acceptance signals
- The computation of Business Failure or Success by detecting if the response document was specified – at design time – with isPositiveResponse=false.
- The occurrence of a Notification of Failure business message - Although not part of or described in the BT patterns, General Exception may occur that impacts a party's capability. The NOF and General Exception are described later in Section 3.6.2.3.

Both parties can compute the Success or Failure of the transaction if reliable messaging, as well as request and response Acceptance Acknowledgement signals, is used. Once Success or Failure is thus established, the Business Transaction is considered closed with respect to both parties. If reliable messaging is not used, state alignment cannot be guaranteed and therefore it could happen that one party believes the transaction has been successful, while the other believes it ended in Failure.

Upon receipt of a response the Requesting Activity MAY send a Receipt Acknowledgement and/or Acceptance Acknowledgement signal back to the Responding role. This operation does not pass control back to the Responding activity. When the Requesting Party send the signal(s) after the defined timeouts occur (Receipt or Acceptance Acknowledgement), the Business Transaction is considered null and void. This may be subject to the agreement of the parties.

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.

3.5.1.1 Transaction Interaction Patterns

The Business Transaction pattern and operational semantics will specify whether a Requesting Business Document requires a Responding substantive document in order to achieve a "Success" end state. In addition, the Business Transaction MAY specify a proper nonzero time duration for TimeToPerform, imposing a deadline for the substantive response. A substantive response is a business message that includes a Business Document rather than a non-substantive Business Signal that MAY or MAY not include identification data.

Furthermore, the specification of a Business Transaction MAY indicate, for the request whether Receipt Acknowledgement and/or Acceptance Acknowledgement are required, and for the response whether Receipt Acknowledgement and/or Acceptance Acknowledgement are required.

The specification of a Business Transaction MAY require each one of these business signals independently of whether the other is required. Therefore there is a finite set of combinations. The ebBP specification supports a subset of all possible combinations based on the patterns defined earlier in this document. The condition guards on state transitions are described in further detail later in Section 3.

Note: In addition to the concrete patterns, the Legacy Business Transaction pattern (known in preceding versions as Business Transaction) is being retained for conversion purposes only. Industry or communities are recommended to define and use the extensible Data Exchange pattern if the process pattern requires specialization.

3.5.2 Business Transactions and Shared Intent

Trading partners MAY wish to indicate that a Business Transaction performed as part of an ebXML arrangement is, or is not, intentional. A declaration of intent to be bound may assist in establishing the equivalence of an electronic message to an enforceable-signed physical writing. Parties MAY create explicit reference of that shared intent when they use the ebBP technical specification by manipulating the parameter ("hasLegalIntent") as described in Section 3.4.9.7.

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). However, documents which rely solely on the presence of a signature MAY or MAY NOT be correctly interpreted, if there is semantic content indicating the conditions the parties expect.

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

The hasLegalIntent parameter occurs at the BusinessTransactionActivity level, which means that the performing of a BusinessTransaction within a Binary (Business) Collaboration is either specified as intentional or not. As specified in Section 3.4.9.7, mechanisms in the BSI provide the capability to support this constraint (or shared intent) such as reliability, document security, non-repudiation, etc. The default value is “false.”

These three descriptions have been extracted from the eCommerce Patterns v1.0 white paper for informational reference (See Section 5 for the white paper location).

- Legally Binding – An optional character of a statement or commitment exchanged between trading partners (such as an offer or acceptance), set by its sender, which indicates that the sender has expressed its intent to make the statement or commitment legally enforceable.
- Non-binding -- An optional character of a statement or commitment exchanged between trading partners (such as an offer or acceptance), set by its sender that indicates the intent to be legally bound. See first description above.
- Trading partners MAY also wish to exchange proposed terms, without making an assertion of intent to be legally bound. This is analogous to the paper contracting practice of exchanging unsigned drafts or term sheets.

3.5.3 Non-Repudiation

Trading partners MAY wish to conduct intentional 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 an obligation, in the case that the counter party 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 in this technical specification. Each protocol provides the user with some degree of additional evidentiary information by creating or requesting additional artifacts that would assist in a later questions over repudiation issues. Neither is a dispositive absolute assurance.

One expects 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., requester 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 or a comparable mechanism be used. So NonRepudiationOfReceipt is meaningless if ReceiptAcknowledgement is not required. Failure to conform to 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 (or comparable mechanism) receipt will satisfy the imposed timeout deadline. Thus, a Failure to send a signed receipt within timeToAcknowledgeReceipt, would make the transaction null and void, i.e. the agreed upon expectations of business significance of the Requesting party has not been adhered to in the activity.

3.5.4 Authorization security

Each request or response MAY be sent by a variety of individuals, representatives or automated systems associated with a business partner.   There MAY be cases where trading partners have more than one ebXML or correspondingly capable BSI, 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 MUST 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 [activity's role] to a list of allowed values previously supplied by that party.

isAuthorizationRequired is specified on the Requesting and Responding Activity accordingly. Authorization typically relates to a signed Business Document and the association to the role identity of the party expected for that activity. Acknowledgement signals MAY communicate authorization failures. It is important to surface exceptions so action can be taken. Some conditions where authorization MAY apply and be related to exceptions include:

- When business rules are applied
- When a communication is persisted
- When a business message is submitted for acceptance processing

Based on agreements, the parties may establish the authorization parameters to establish these capabilities. If authorization is enabled, the Business Document and Business Signal SHOULD be authenticated or tamper detection enabled. In this version, the mechanisms for a BSI 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, is undefined. This quality of service attribute is like a hint to the BSI and MAY be delegated to an underlying service.

In this version, the mechanisms for a BSI 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, is undefined. This quality of service attribute is like a hint to the BSI and MAY be delegated to an underlying service.

3.5.5 Document security

The value of isConfidential, isTamperDetectable, and isAuthenticated apply to the Document Envelope (primary logical Business Document) or Attachment. 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.

- The communications channel used to transport the Message provides transient authentication. The specific method will be determined by the communications protocol used.
- Persistent authentication means the Business Document signer’s identity MUST be verified at the receiving application level. Authentication assists in verification of role identity of a participating party.
- Transient confidentiality is provided by a secure network protocol, such as SSL as the document is transferred between two adjacent ebXML Messaging Service (MSH) or other transport messaging nodes.
- Persistent confidentiality is intended to preserve the confidentiality of the message such that only the intended party (application) can see it. The message MUST remain in encrypted form after it is delivered to the messaging node and will be decrypted only by the authorized application. S/MIME MAY 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 messaging node, between the messaging node and the application. Tamper detection assists in verification of content integrity between and within a participating party.

As with reliability, the parties may establish the assurance parameters, for example. The level of document security (i.e. the documentSecurity attribute group used) of Business Documents or Attachments SHOULD adhere to the operational semantics held in the BT pattern matrices.

Agreements may also be relevant to establishing these capabilities (See earlier subsections in Section 3 for further detail). If non-repudiation of content is required, these attributes SHOULD be enabled (i.e. the enumeration selected for each of these values is other than 'none.'). Typically, this occurs in intentional situations where authentication and tamper detection are particularly important to support enforceability. In such cases, the parties SHOULD also specify the channel is confidential (i.e. this practice is recommended). Otherwise, the parties involved specify document security. See the patterns matrices earlier in Section 3 for other details. In those instances where intent is specified regardless of pattern, documentSecurity attributes apply. For example, where non-repudiation of content is required, documentSecurity should apply although this is subject to the agreement of the parties. Updates to documentSecurity MAY also be made in the CPA.

3.5.6 Reliability

The parameter isGuaranteedDeliveryRequired at the Business Transaction level states whether guaranteed delivery of the transaction 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.

3.5.7 Parameters required for CPP/CPA

The ebBP technical specification provides parameters that can be used to specify certain levels of security and reliability. This specification provides these parameters in general business terms.

These parameters are generic requirements for the business process, which may be used ebXML or hybrid (ebXML and web services) implementations. These parameters MAY be 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 securities, and reliability above, are determinants 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.

3.5.7.1 Handling Partner Roles

The CPP and CPA also use the roles defined for a party in the Business Collaboration that map to corresponding ones in the CPP or CPA. The Business Collaboration provides a general prescription of the roles a business partner can play. A trading partner may play multiple roles and are specified in the CPP or CPA.

The mapping of the roles to the Business Transaction MAY vary between Business Collaboration instances. Roles MAY also map differently in a Business Collaboration instance. For example, in an CPA negotiation, a trading partner may be a requester or responder in the same Business

Collaboration. Translating that to a CPA, the trading partner can serve both (or multiple) roles of Requester and Responder. In the negotiation example, the role mapping of the trading partner as requester MAY be relevant to the role mapping when the same trading partner acts as the responder.

In the CPA, this is handled by the choreography that includes specific Business Transaction Activities. For example, the trading partner acts as the Initiator for the sending of another offer. In another BTA, the same trading partner is the responder. In this example, the choreography should be explicit about this transition. Each defined BTA would relate to a separate Performs (in the ebBP schema) even though the role reference remains the same. These constructs allow role mapping across Business Collaborations, activities and BT.

3.5.7.2 Handling Operation Mapping

In CPA and WSDL, service context SHOULD be concrete and MAY map to the business services abstractly defined in the ebBP schema. In the CPA, extensions SHOULD be used to identify a concrete web service (WSDL) endpoint. Where the relationship is explicit, the Action Context SHOULD be used to map the web services endpoint identified in CPA to the corresponding BTA through the abstract operation (WSDL) name in the ebBP schema.

Where the ebBP schema is used but the OperationMapping is not explicitly defined, the business partners SHOULD manage the service mappings. Through a business service, the OperationMapping MAY also support Business Transactions defined in other than XML where different identification mechanisms are used. This allows the binding of service and business endpoints.

3.6 Run time Business Transaction Semantics

The ebXML concept of a Business Transaction and the semantics behind it are central to providing predictable and supporting enforceable commerce. It is expected that any BSI will be capable of managing a transaction according to these semantics.

Therefore, the BSI, or any software that implements one role in an ebXML Business Collaboration, SHOULD at minimum to be able to support the following transaction semantics:

- Detection of the opening of a transaction
- Detection of transfer of control
- Detection of successful completion of a transaction
- Application of business rules expressed as schema definitions and isPositiveResponse or isPositiveSignal for determination of Success
- Detection of failed completion of a transaction
- Detection of timeouts
- Detection of protocol exceptions
- Validation of the received response or signal and identify if it was specified with isPositiveResponse = false or adherence to the fixed isPositiveSignal value
- Detection of Business Failures (such as Notification of Failure)

ebXML does not specify how these transaction semantics are implemented but it is assumed that any BSI will be able to support these basic transaction semantics at runtime. If either party cannot provide full support, then the requirements MAY be supported by or relaxed as overrides in the CPP or CPA.

The following sections discuss the two causes of Failure: timeouts and exception. When either one happens, typically and as unless otherwise agreed by the parties, 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 technical mechanisms used for compensation is outside of the scope of this technical specification. The responsibilities of the two roles differ slightly and are described in each of the sections below. When a Failure other than a timeout occurs at either the Responding or Requesting role, an exception signal or Notification of Failure business message MAY be sent based on the circumstances and the parties' defined expectations. If used, typically both parties will exit the current Business Transaction. The Notification of Failure is explained in Section 3.6.2.3.

3.6.1 Timeouts

Since all Business Transactions must have a distinct time boundary, there are timeout parameters associated with the Response and each of the acknowledgement Business Signals (Receipt and/or Acceptance). If Business Signals and/or a Response apply in the BT used and a timeout occurs before the corresponding Response or Business Signal arrives, the transaction MUST be 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 request is sent and the substantive response is received.

Table 8 Timeout Parameters

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 party expects Business Signals in return to the Business Document Request or Response. A Requesting party 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. Correlating timeouts is partner-specific. All timeouts typically SHOULD be reported independent of their priority. The timer values MUST conform to the well-formedness rules for timer values. Refer to Section 3.8.

When used, a BSI SHOULD adhere to 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 party simply terminates if a timeout is thrown. This prevents Responding Business Transactions from hanging indefinitely.

The total time allowed for a BTA to complete is therefore, TimeToPerform that is equal to or greater than the larger of timeToAcknowledgeReceipt and the timeToAcknowledgeAcceptance on the Request plus the TimeToPerform that is equal to or greater than the larger of the larger of timeToAcknowledgeReceipt and the timeToAcknowledgeAcceptance on the Response (given which, if any, are used).

The timeToAcknowledgeReceipt is the duration from the time a Business Document in a Requesting Activity is sent by a Requesting party until the time a verification of receipt is properly received by the Requesting party. The time to acknowledge business acceptance of a Requesting Business Document is the duration from the time a Requesting party sends a Business Document until the time an Acceptance Acknowledgement Business Signal (non-substantive) is properly received by the Requesting party from the Responder.

Timing parameters or expectations MAY change during the Business Collaboration lifecycle, and conditionality exists where late binding constructs MAY be used. For example, in telecommunications timing may be renegotiated during execution.

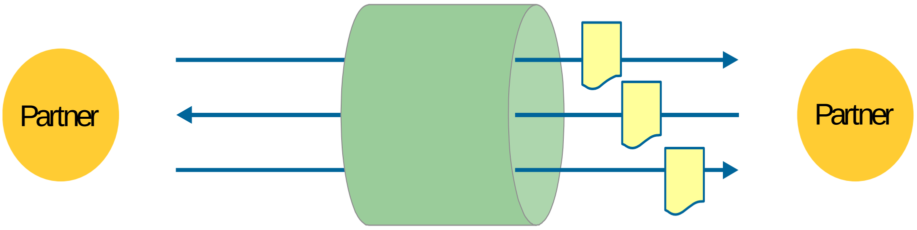


Figure 12: Timing Changes in Process Lifecycle

Actually timing MAY be handled in these parameters or in the choreography. In the latter, the timing requirements are specified in different activities defined in the choreography (for example, delivery).

A Variable MAY be used to allow the flexibility. Variables were described in more detail in Section 3.4.11.1, Key Semantics of Choreography. A Variable MAY have a duration, a type and, where applicable, a default value. Variables MAY also be specified externally and the value acquired.

3.6.2 Protocol Exceptions

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

3.6.2.1 Receipt Acknowledgement Exception

A Receipt Exception signals an error condition in the management of a Business Transaction. This Business Signal is returned to the initiating activity that originated the request. This exception MUST terminate the Business Transaction. These errors deal with the mechanisms of message exchange such as verification, validation, authentication and authorization and will occur up to message acceptance. Typically the rules and constraints applied to the message will have only dealt with the well-formedness 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 BTA. Note that the receiving BSI can only identify this exception.

- 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.

3.6.2.2 Acceptance Acknowledgement Exceptions

An Acceptance Exception signals an error condition in a Business Activity. This Business Signal is returned to the initiating role that originated the request. This exception MUST terminate the Business Transaction. These errors deal with the mechanisms that process the Business Transaction and will occur after message verification. Typically the rules and constraints applied to the message will deal with the semantics of message elements and the validity of the request itself. This exception MAY also apply when 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 unable 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 Business Signals, or timeout 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 Acknowledgement Exception signals the exception back to the Responding role and terminates the Business Transaction.

3.6.2.3 Notification of Failure Business Messages and General Exception Signals

A Notification of Failure business message is a choreographed behavior that is defined (i.e. planned for use where necessary). Conversely, if specified by the parties, the General Exception signal MAY handle unchoreographed/unplanned events (unforeseen and, most often, catastrophic in nature) for a party when that party is in control during a Business Transaction. If agreed amongst the parties, any BSI at any point MAY issue the Notification of Failure business message in time, during, or after a collaboration. The Notification of Failure is not intended to be reported by Receipt and Acceptance Acknowledgement Business Signals, especially when one of the party (typically the Requesting party) is not in control of the Business Transaction protocol or between BTAs.

> *Implementation Note:*
>
> *Additional operational semantics may exist in the patterns matrices rather than being held in the ebBP schema. For example, manual or implicit actions by an involved party may be relevant in the ebBP process definition, particularly to provide state transition information in the collaboration for monitoring. In the appendices to this technical specification, a brief description is provided about how the patterns may be used when*

> *manual or implicit actions exist. In future versions, more semantics may be defined and included in the ebBP technical specification and/or schema as business requirements are identified or user community feedback received.*

The Notification pattern is a formal exchange and requires non-repudiation. When the Notification of Failure is used (for the Notification pattern), a Business Transaction MUST be set aside. A separate communication channel is recommended. If defined by the parties, the NOF MAY occur:

- After timeout occurs on receipt of a response, NOF
  - MAY occur for Failure to receive a Requesting or Responding Business Document
- When a party has conditional acceptance or when the party can't determine that condition (i.e. no response received at timeout on Time To Perform)
- When a party is not under control (differentiates from General Exception)
- When an offer is made and needs to be rescinded as the transaction failed (Business Failure)
- If a timeout occurs and no/no more retries are available (and TTP has not expired). If retries still exist and a timeout has occurred, the offeror can choose to retry or send a Notification of Failure

NOF does not rely on the EndsWhen related to a Business Activity. In the cases such as those above, the transaction is set aside.

Generally if a business retry is initiated and a response received, the latter can be used. If this occurs, the parties will be responsible for identifying and dealing with duplicate business messages (in this case a duplicate request). Duplicate elimination logic SHOULD reject the business retry, and possibly resend the business response, which would then also be recognized as a duplicate. This allows the sender to process the original response safely and mitigate the overhead to wait for the response to a business retry. This could also improve efficiency, lowering the need for backend systems support.

The business retry for a RequestingBusinessActivity identifies the number of retries allowed in addition to the initial request while the Time To Perform has not been exceeded. The business retry MAY be associated with control exceptions such as timeouts. If the number of retries is not specified, the parties have not agreed to use a business retry. The Requesting party may retry as many times as they choose (i.e. it is not constrained to a specific number). If a business retry count of 3 is chosen (in addition to the initial request), the Requesting party MUST retry up to 3 times (i.e. until a retry is successful as long as the retry count has not been exceeded). Business retries SHOULD NOT apply to Exception signals.

For example, if a business retry was not specified and a response was not received, an NOF could be issued. If the response is received, it is then ignored because the NOF has negated the Business Transaction. In the future after more business requirements are gathered, the business retry count will be further specified in relationship to the choreography.

It is recognized that NOF and the specific details/requirements should be primarily driven by the agreements between business partners. One possible scenario example could involve the issuance of a General Exception signal (business control Failure) by a Responder and NOF (stop transaction) by Requester. Responder exits a transaction, and uses a business control Failure (which MAY equate to a Negative Receipt Acknowledgement, Acceptance Acknowledgement, or General Exception signal). The Requester MAY in turn, issue the NOF.

Typically, in the case when there is reliable messaging which shows the receipt of request or response, the party MAY not be capable of or required to send a NOF. If for example, a response is sent then a NOF by a Responder. That is actually an anomaly and MAY be handled by the agreement of the parties.

The General Exception signal MAY be used under other conditions such as:

- isIntelligibleCheckRequired exists and a Receipt Acknowledgement has been sent, but something fails in processing. This is assuming that an Acceptance Acknowledgement is required, processing has begun but not completed, and the AA has not yet been sent.
- isIntelligibleCheckRequired has not been defined and a ReciptAcknowledgement has been sent, but something fails in processing. An AcceptanceAcknowledgement may or may not be required later.
- No signals are required and the need exists to notify a business partner of a problem. This could support the known RosettaNet case of synchronous events.

The key is that the technical failure be visible for sufficient state resolution. For example, an unexpected gateway shutdown may require a General Exception signal be issued. Under these circumstances, an event outside of the collaboration (gateway shutdown) impacts it (collaboration).

A General Exception is a limited case and distinct type of technical failure, i.e. AnyProtocolFailure. The involved parties determine if such exceptions are used in order to recognize and handle the possibility of a catastrophic failure.

As an unchoreographed event, a General Exception MAY result in later actions of the parties that are choreographed. A General Exception MAY result in a state transition to a technical failure (AnyProtocolFailure). Similar to other technical failures such as the Receipt Acceptance Acknowledgement Exceptions, AnyProtocolFailure is designed to allow the protocol to catch and handle behavior when the protocol fails because of technical failure. Note, state transitions and failures are described earlier in Section 3 and in more detail in Section 3.6.3. If a General Exception occurs and the party notifies the other with a General Exception signal, the parties transition to a known state. Whether further action is required or the technical failure results in any business effect is subject to the agreement of the parties.

Should a General Exception not be defined between the parties, i.e. there is no mechanism defined to handle such events, the parties MAY use alternate means or act in line with any agreements between them.

Under choreographed circumstances, if a party is unable to respond with a choreographed Receipt Acknowledgement within the time specified, that party SHOULD exit and, if agreed by the parties, the Requesting party MAY issue an NOF or a business retry. For the unchoreographed General Exception, the parties MAY also agree to subsequent actions that are choreographed.

Whether the unchoreographed General Exception follows the same path as the known circumstances outlined is unspecified.

> *Implementation Note: The General Exception is outside of the currently defined concrete BT patterns. Software implementers MAY choose to enable software that is aware of this Exception type.*

Should a NOF business message be specified by the parties but not sent after an Exception, another Protocol Failure (choreography violation) SHOULD occur. More business requirements are sought to understand, if and when an NOF should be issued, another Business Transaction may occur after the return to initial state, or subsequent choreographed actions are required.

In addition, more business requirements are being sought to understand needs regarding propagation of errors in complex activities such as Business Collaboration involving more than two parties and in a ComplexBTA. The same holds true for the business retry count and further specification of it in relationship to the choreography. When the business retry is used, the time to Acknowledge Receipt and/or Acceptance (given which are used) SHOULD be reset although the TimeToPerform SHOULD NOT. Process (signal) timeouts are recoverable within retry parameters and not recoverable outside of the retry parameters.

3.6.2.4 BSI Conformance

In order to produce the appropriate exceptions, the BSI MUST conform to the following parameters. The Requesting and Responding roles take different action as per below.

isAuthorizationRequired

If a business partner role needs authorization to request a Business Action or to respond to a Business Action then the sending party role MUST sign the Business Document exchanged and the receiving party role MUST validate this business control and approve the authorizer. A Responding party MUST signal an authorization exception (Receipt Exception) if the role of the Requesting party role is not authorized to perform the Business Activity. A sending (Requesting) party MUST send notification of failed authorization if a requesting party 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 Party MUST signal a Receipt Exception if the sending (Requesting) party role has not properly delivered their Business Document. Similarly, a requesting party MUST send Receipt Exception if a Responding party has not properly delivered their Business Document.

***isNonRepudiationOfReceiptRequired*.**

Both business partners agree to mutually verify receipt of a Requesting Business Document and that the receipt MUST be non-reputable. If agreed by the parties to use NOF, a Requesting party MUST initiate a Notification of Failure Business Transaction if a Responding party has not properly delivered signed their receipt. For a further discussion of non-repudiation of receipt, see also the ebXML E-Commerce and Simple Negotiation Patterns (See references at the end of this technical specification).

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

A parameter that MAY take the value of TRUE or FALSE. This is a Boolean attribute. If TRUE this DocumentEnvelope is intended as a positive Response to the Request. If isPositiveResponse = FALSE, the BTA ends in Business Failure mode. The value for this parameter supplied for a DocumentEnvelope is an assertion by the sender of the DocumentEnvelope regarding its intent for the transaction to which it relates, but does not bind the recipient, or override the computation of transactional Success or Failure.

3.6.3 Computation of the status of a Business Transaction Activity

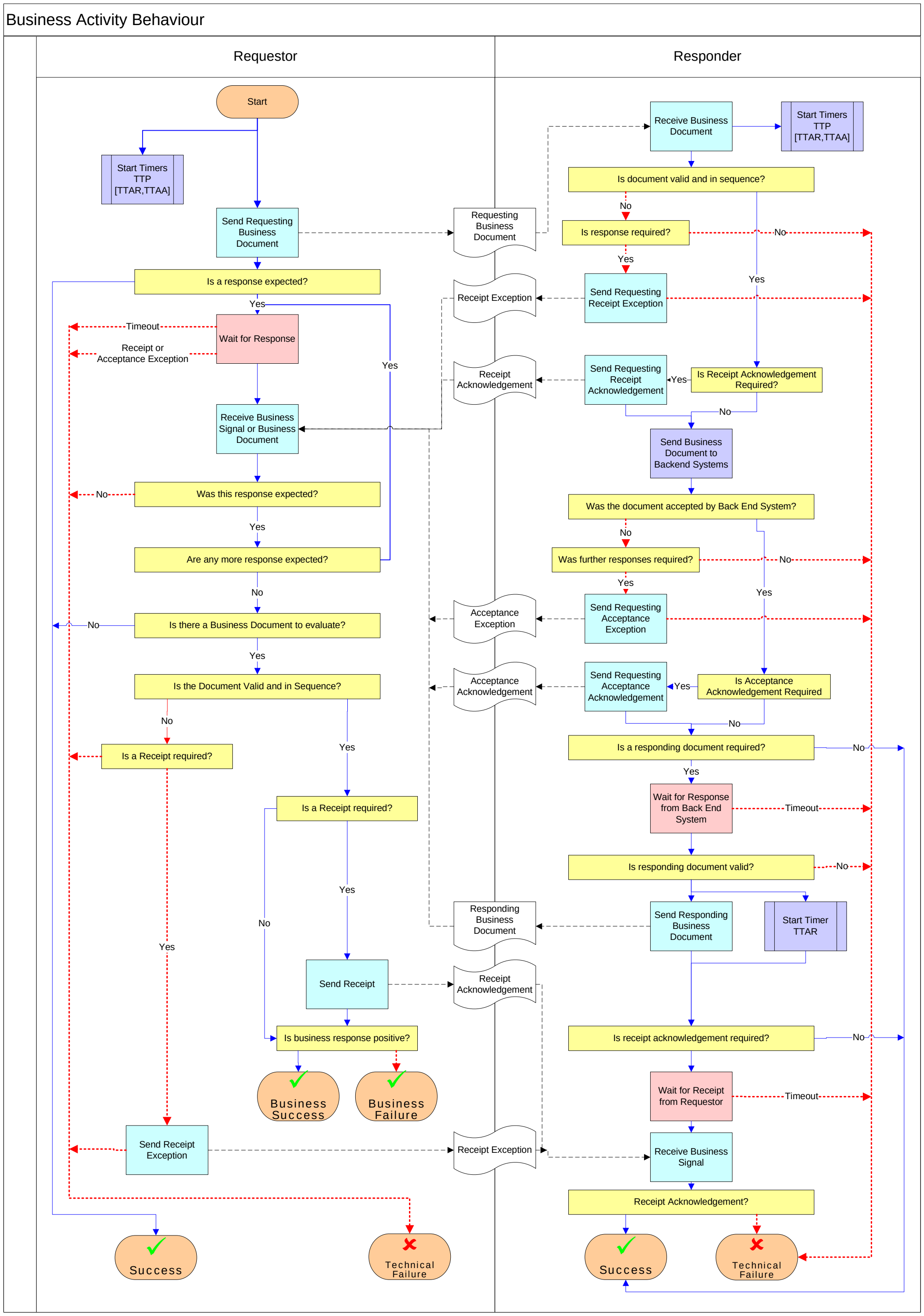

Business Activity Behaviour
Requestor
Responder
Start
Start Timers TTP [TTAR,TTAA]
Send Requesting Business Document
Requesting Business Document
Is a response expected?
Yes
Wait for Response
Timeout
Receipt or Acceptance Exception
Yes
Receive Business Signal or Business Document
Was this response expected?
No
Yes
Are any more response expected?
No
Is there a Business Document to evaluate?
No
Yes
Is the Document Valid and in Sequence?
No
Yes
Is a Receipt required?
Yes
Is a Receipt required?
No
Yes
Send Receipt
Is business response positive?
Business Success
Business Failure
Send Receipt Exception
Success
Technical Failure
Receive Business Document
Start Timers TTP [TTAR,TTAA]
Is document valid and in sequence?
No
Is response required?
No
Yes
Yes
Send Requesting Receipt Exception
Receipt Exception
Send Requesting Receipt Acknowledgement
Receipt Acknowledgement
Yes
Is Receipt Acknowledgement Required?
No
Send Business Document to Backend Systems
Was the document accepted by Back End System?
No
Was further responses required?
No
Yes
Yes
Send Requesting Acceptance Exception
Acceptance Exception
Send Requesting Acceptance Acknowledgement
Acceptance Acknowledgement
Yes
Is Acceptance Acknowledgement Required
No
Is a responding document required?
No
Yes
Wait for Response from Back End System
Timeout
Is responding document valid?
No
Send Responding Business Document
Responding Business Document
Start Timer TTAR
Receipt Acknowledgement
Is receipt acknowledgement required?
No
Wait for Receipt from Requestor
Timeout
Receipt Exception
Receive Business Signal
Receipt Acknowledgement?
Success
Technical Failure

Figure 13: Computation of the Status of a Business Transaction Activity

As described in this section, Figure 13 represent the computation of the Success or Failure of a BTA based on the different possible scenarios. Note that this diagram (for brevity) does not specify the use of an Acceptance Acknowledgement Business Signal on the Response or business retries (related to retryCount). A General Exception signal could also be used if a scenario dictates its use, as indicated in a previous Section 3.6.2.3.

The values of the enumeration of the state of a Business Transaction of a condition guard on a transition are:

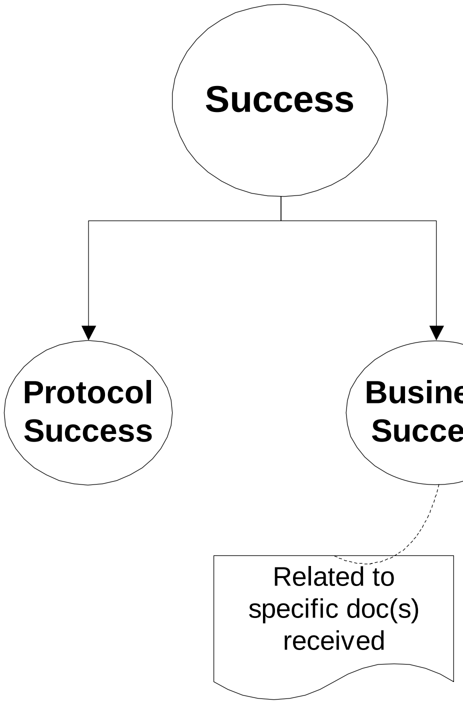
- ProtocolSuccess
- AnyProtocolFailure
- RequestReceiptFailure
- RequestAcceptanceFailure
- ResponseReceiptFailure
- ResponseAcceptanceFailure
- SignalTimeout
- ResponseTimeout
- BusinessSuccess (isPositiveResponse=true or no isPositiveResponse attribute)
- BusinessFailure(isPositiveResponse=false)
- Success (Both Protocol and Business Success)
- Failure (AnyProtocolFailure or BusinessFailure)

Each of the defined Business Transaction states of a condition guard that relate to failures in essence has a handler (or interface). For example, AnyProtocolFailure defines transition to that handler associated with a technical failure. Two tree diagrams are provided to assist in understanding and using these state transitions, Figure 14 showing a successful path and Figure 15 specifying Failure. In addition to a corresponding Figure 11 showing successes and failures, each tree diagram provides useful views into different relations that are represented. For example, Business Success and Business Failure relate to the Business Documents received. While conversely, any timeout is a business Protocol Failure, i.e. the state is not aligned.

The enumerated state values represent:

For Success:

- Success (Both Protocol and Business Success)
- ProtocolSuccess: Technical Success. For example, acknowledgement of receipt signal received for a Request prior to a timeout.
- BusinessSuccess (isPositiveResponse=true or no isPositiveResponse attribute): Specific document(s) are received.



2800

2801 **Figure 14: 'View' of Success**

For Failure:

- Failure (AnyProtocolFailure or BusinessFailure): For example, specific document(s) are received.
- BusinessFailure (isPositiveResponse=false): Specific document(s) are received.
- AnyProtocolFailure: Technical failure such as those specified or any other
    - Note: As previously indicated, General Exception is a distinct case of the technical failure called AnyProtocolFailure.
- ResponseTimeout: Time to Perform exceeded.
- SignalTimeout: Time to Receipt or Acceptance Acknowledgement exceeded.
- RequestReceiptFailure: Technical failure of Receipt Acknowledgement on Request.
- RequestAcceptanceFailure: Technical failure of Acceptance Acknowledgement on Request.
- ResponseReceiptFailure: Technical failure of Receipt Acknowledgement on Response.
- RequestAcceptanceFailure: Technical failure of Acceptance Acknowledgement on Response.

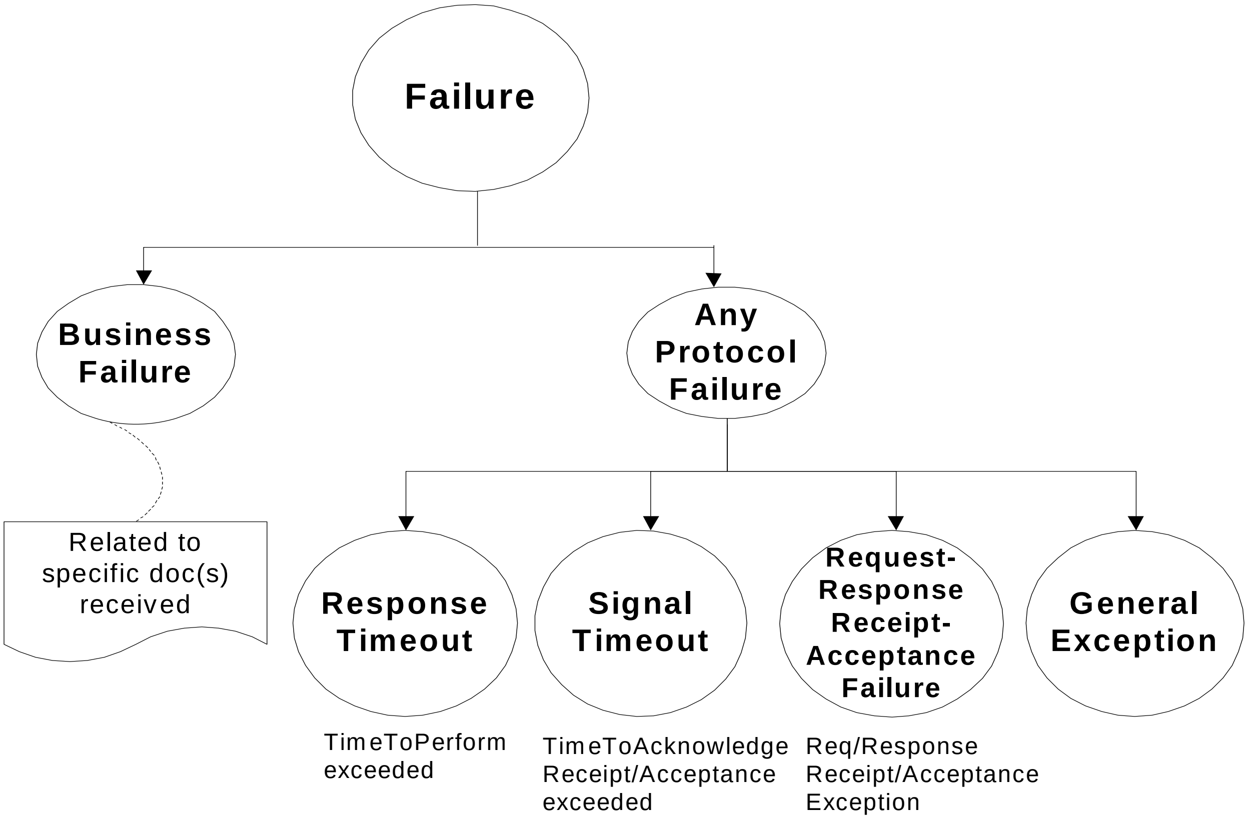

Failure
Business Failure
Any Protocol Failure
Related to specific doc(s) received
Response Timeout
Signal Timeout
Request-Response Receipt-Acceptance Failure
General Exception
TimeToPerform exceeded
TimeToAcknowledge Receipt/Acceptance exceeded
Req/Response Receipt/Acceptance Exception

Figure 15: 'View' of Failure

In real-world scenarios, it is anticipated that more than one condition guard MAY occur and the parties involved MAY choose to monitor them. Monitoring can continue even if an initial Failure or timeout has occurred. The affected parties are notified as soon as possible.

Transitions exist with guards. When more than one condition guard is defined (by the parties), they MAY be mutually exclusive or all used. If not defined, the assumption is all MAY happen. For example, SignalTimeout will occur before ResponseTimeout.

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 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 is to allow the definition two (or more) “logical” documents from the same physical document and a Condition Expression evaluated at runtime by the BSI. If the condition is true and isPositiveResponse = false, then the transaction ends in BusinessFailure 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.

Each BTA MUST be designed such that there is at a minimum two transitions from the BTA, one with a condition guard 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.

3.7 Where the ebXML Business Process Specification May Be Implemented

The ebBP technical specification SHOULD be used wherever software components are being specified to perform a role in an ebXML Business Collaboration. Specifically, this technical specification is intended to provide the business process and document specification for the formation of ebXML trading partner Collaboration Protocol Profiles and Agreements.

However, the ebBP technical specification MAY be used to specify any eCommerce, eBusiness or shared 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.

The Operation Mappings allow for using the ebBP technical specification and schema for mapping web service interactions without any other required ebXML support such as Collaboration Protocol Profile or Message Service (although they could be used). The ebBP technical specification allows the definition of pure message exchange in a choreography including constructs for state alignment using Business Signals, state transition and condition guards, etc.

3.8 Business Collaboration and Business Transaction Well-Formedness Rules

3.8.1 Assumptions

XInclude processing and AttributeSubstitution processing MUST be performed prior to both schema validity and well-formedness checks.

> *Implementation Note*
>
> *It is the responsibility of designers using XInclude for file and package modularity to ensure that any collisions of ID values are removed using AttributeSubstitution.*

Also implementers are reminded that the IDREFs SHOULD be changed to reflect the new collision-free ID values.

Elements in the ebBP instance MUST be uniquely identifiable from outside of that instance. Therefore, a qualified identifier syntax is not required. The nameID is document-scoped, irrelevant of package structure. The benefit of using a document-scoped identifier is that the processor of the referring document requires no semantic knowledge of the referred-to document. The focus is on the identification of elements within the ebBP instance.

In the majority of cases (and as supported by ebBP schema), the name and nameID SHOULD be required, and serve different functions for the user community. The Name attribute SHOULD NOT be used, nor is it intended, for referencing, although it may be important to the business analyst.

3.8.2 Referential Constraints

[Package/@parentRef]

The @parentRef attribute's value MUST be a value of a @nameID attribute of a Package.

[AttributeSubstitution/@nameIDRef]

The @nameIDRef attribute's value MUST be a value of a @nameID attribute (with type ID) of some element.

[DocumentEnvelope/@businessDocumentRef]

Every @businessDocumentRef attribute's value MUST be a value of a @nameID attribute of a BusinessDocument.

[Attachment/@businessDocumentRef]

Every @businessDocumentRef attribute's value MUST be a value of a @nameID attribute of a BusinessDocument.

[BusinessTransactionActivity/@businessTransactionRef]

Every @businessTransactionRef attribute's value MUST be a value of a @nameID attribute of an element in the substitution group of BusinessTransactionHead. [Note: These elements MAY be children of ProcessSpecification or children of Package.]

[CollaborationActivity/@collaborationRef]

Every @collaborationRef attribute's value MUST be a value of a @nameID attribute of either a BusinessCollaboration, a MultiPartyCollaboration, or a BinaryCollaboration.

[Note: New business process definitions SHOULD use BusinessCollaboration as the basic Collaboration Activity unit of reference.]

[FromLink/@fromBusinessStateRef]

Every @fromBusinessStateRef attribute's value MUST be a value of a nameID attribute of either a BusinessTransactionActivity, a CollaborationActivity, or a ComplexBusinessTransactionActivity. Each of these elements referred to MUST be in the same Collaboration elements that the FromLink is in (that is, MUST be siblings with either a BusinessCollaboration, MultiPartyCollaboration, or BinaryCollaboration parent).

[ToLink/@toBusinessStateRef]

Every @toBusinessStateRef attribute's value MUST be a value of a @nameID attribute of either a BusinessTransactionActivity, a CollaborationActivity, or a ComplexBusinessTransactionActivity. Each of these elements referred to MUST be in the same Collaboration elements that the ToLink is in (that is, MUST be siblings with either a BusinessCollaboration, MultiPartyCollaboration, or BinaryCollaboration parent).

[Performs/@currentRoleRef]

Every Performs element MUST have one @currentRoleRef attribute whose value matches the value of a @nameID attribute on a previously mentioned Role element.

Note: Role elements are mentioned at the top-level of a ProcessSpecification (within the ExternalRoles element), and then in each Collaboration element (BusinessCollaboration, MultiPartyCollaboration, BinaryCollaboration) that is not an inner collaboration. After these contexts, roles are introduced in additional Collaborations that are referenced within a CollaborationActivity element.

[Performs/@performsRoleRef]

Exactly one of @performsRoleRef MUST be present under Performs. When @performsRoleRef is used, its value MUST be a @nameID value of a Role element that is declared in the next Collaboration context. From a BTA, the @nameID value in the @performsRoleRef attribute must match either the @nameID value of RequestingRole or RespondingRole in the BT.

There must be two Performs, and they must reference different Role elements in the BT (that is, one must match value of RequestingRole/@nameID and the other must match value of RespondingRole/@nameID.

Note: When a Role/@nameID is the same in both the current and the next Collaboration context, it is assumed to be the same Role, and so the Performs association is not needed. Performs is needed for Role switching (that is, when a participant that had been a buyer, now reenters the collaboration as a seller), to match up roles differing in names in, for example, included packages, and as needed elsewhere. The core schema constrains when the Performs element is not required.

[@signalDefinitionRef]

Specializations (elements of the substitution group) of BusinessTransaction contain RequestingBusinessActivity and RespondingBusinessActivity elements whose content models MAY contain child elements whose types are subtypes of SignalEnvelopeType. The @signalDefinitionRef attributes of these child elements MUST have values of a @nameID value of a Signal element of type SignalType.

[Variable/@businessDocumentRef]

Every @businessDocumentRef attribute's value MUST be a value of a @nameID attribute of a BusinessDocument

[Variable/@businessTransactionActivityRef]

Every @businessTransactionActivityRef attribute's value MUST be a value of a @nameID attribute of a BusinessTransactionActivity.

[OperationMapping/@roleRef]

Every @roleRef attribute's value MUST be a value of a @nameID attribute of a Role element contained in either a BusinessCollaboration, MultiPartyCollaboration, or BinaryCollaboration.

[OperationMapping/@businessTransactionRef]

Every @businessTransactionRef attribute's value MUST be a value of a @nameID attribute of an element in the substitution group of BusinessTransactionHead.

[MessageMap/@documentEnvelopeRef]

Every @documentEnvelopeRef attribute's value MUST be a value of a @nameID attribute of a DocumentEnvelope.

[SignalMap/@documentEnvelopeRef]

Every SignalMap@documentEnvelopeRef attribute's value MUST be a value of a @nameID attribute of a Signal.

3.8.3 Functional or Other Well-Formedness Rules

3.8.3.1 Specification Element

- When a Specification element is optional on a Business Document element, this indicates that the Business Document is abstract and substitution can be used to replace the logical Business Document with a physical one that is relevant to a particular domain or use.
- Inclusion: Only packages MAY be used with the XInclude mechanism.
- A user is responsible to understand where to include packages that are valid when XInclude mechanism is used.

3.8.3.2 Variables

- When the Variable element is used, it is cast in a type that is usable in that ConditionExpression.
- Any variables used in the condition tree for the BTA guard MUST precede the guard in the execution of the BTA.
- When multiple condition expressions are used, the languages MUST be distinct and the expressions MUST be equivalent (i.e. different from others in the sequence). Expression of conjunction or disjunction is undefined and therefore places responsibility for that function on the expression language.

3.8.3.3 Business Collaborations

- All non-isInnerCollaboration Collaborations (any type of Business Collaboration) are eligible to start another complex Business Collaboration (Binary or Multiparty).
- An outer collaboration TimeToPerform MUST be no shorter than the time of the longest inner collaboration.
- The TimeToPerform duration of a Fork cannot be less that any TimeToPerform duration of its Business Activities.
- When set to ‘true’, the waitForAll attribute of the Join MUST indicate that all transitions coming into the Join MUST be executed for the collaboration to reach the Join linking state (AND-Join), by default, the Join is an AND-Join. Further explanation is found in Section 3.4.11.1.
- Within any Business Collaboration, there MUST be at least one state defined. A state is a BTA, ComplexBTA, or CollaborationActivity (i.e. no stateless collaborations).
- A Collaboration Activity can transition to any type of Business Collaboration.
- When a BTA refers to a Business Transaction, this requires use of an IDREF that belongs to a Business Transaction.
- Links (FromLink/ToLink) SHOULD NOT reference linking constructs.
- Linking constructs MUST reference states in collaboration (Start, Transition, Fork, Join, and Decision).
- An XOR Fork MUST be followed with a Join where waitForAll = false.

3.8.3.4 Business Signals

- If a Business Signal (other than an Exception signal) is received and it is neither in the identified pattern nor in the Business Transaction protocol, it MUST be discarded. Therefore, this constraint does not apply to Exception signals.
- When a Business Signal is included with a Response or a Response received (and signal has not been received), the path taken depends on the use cases fulfilled by the Business Signal. When a business signal fulfills non-repudiation of receipt requirements, it MUST not be contained in the Response. The non-repudiation MAY be handled at the messaging layer, based on the implementation and business partner parameters defined. Other conditions MAY also be handled in the messaging layer.
- If a Negative Receipt Acknowledgement or Negative Acceptance Acknowledgement occurs, no business retry SHOULD occur.
- Where the defined Business Signals are used, the xlink:href attribute of the xlink.grp attributeGroup SHALL have a value that is an URI that conforms to [RFC2396].
- When creating a Business Signals instance based on the ebBP Business Signals schema, the "name" attribute MUST be set to the same value as name attribute for the corresponding ProcessSpecification element within the ebBP instance. For the ebBP instance, this is the @name attribute of the “name” attributeGroup of the root Process Specification element.

3.8.3.5 Roles

- A Performs element MAY be omitted in Collaboration Activities if the same value of roles are involved and only two top-level roles are used.
- A Performs element MAY not be omitted from Business Transaction Activities. This provides for discrete role declaration at the BTA layer. It maps the "role-as-defined-in-

collaboration" to the "role-as-defined-in-transaction" and provides discrete declaration of roles for the Business Collaboration.

3.8.3.6 Notation for Visual Representation

- ebBP does not preclude generating an XML artifact from an Unified Modeling Language (UML)™ model. This technical specification has used the BPMN notation to visualize Business Collaborations.
- When modeling ebBP Business Collaborations in BPMN compensation SHOULD NOT be used.
- Any changes that are identified may result in new changes for the UN/CEFACT Modeling Methodology (UMM).

3.8.3.7 Timing Parameters

- If both are used, timeToAcknowlegeReceipt < timeToAcknowlegeAcceptance.
- If the Acknowledgement Acceptance is not used, the Time To Perform MUST be equal or greater than timeToAcknowlegeReceipt.
- If either or both of timeToAcknowlegeReceipt or timeToAcknowlegeAcceptance are used, the Time To Perform MUST be other than zero.
- timeToAcknowlegeReceipt MUST be other than zero when non-repudiation of receipt is required.
- The Time To Perform MUST be other than zero.
- Where used, the timeToAcknowlegeReceipt and timeToAcknowlegeAcceptance, in conjunction with the Time To Perform MUST be specified for both the Requesting and (when used) Responding Business Activities.

Note: Where large numbers of Business Collaborations are constructed, consistency and completeness may be important in these rules and their use across all business processes. In those cases, other conditions could apply. For example, if non-repudiation is required at the Requesting Business Activity, a Responding Business Document may be required. Typically, process integrators or developers may develop such conditions to bound business completeness across all processes within a particular domain or industry.

3.8.3.8 Operation Mapping

- When an OperationMapping is defined for a BTA, all message interchanges of the BTA including signals MUST be mapped. Abstract operations MAY come from different interfaces in the mapping of a BTA.

3.8.3.9 Other

- In this technical specification, white space is not controlled but implementers may trigger faults or exceptions.
- For the core schema, the Documentation element MUST be the first element of its container.
- ebBP does not preclude generating another XML artifact from its ebBP definition.

4 ebXML Business Process Specification Schema

This technical specification is supplemented by normative appendices as a part of the Spec package. These appendices are intended to be used with the v2.0.4 technical specification.

- Appendix A: An overview of the Business Service Interface
- Appendix B: Relevant CPA-ebBP mapping. Note see the non-normative examples package for instances relevant to this mapping.
- Appendix C: An overview on manual or implicit activities
- Appendix D: An overview of recursive or optional activities
- Appendix E: ebBP Glossary
- Appendix F: Acknowledgements
- Appendix G: Revision History

Exemplary signal and process definition instances are found on the OASIS web site. This package is separate as more examples are anticipated as more user communities and interested parties use ebBP.

Other non-normative information is provided as indicated earlier in this technical specification.

The previous section provides well-formedness rules relevant to this technical specification and ebBP schemas (core and Business Signals). Note, that the schema syntax is consistent with this technical specification, whereby the latter specifies the conformant capabilities (MUST, SHOULD or MAY for example). The schemas and their associated documentation, and this technical specification are used together.

4.1 Documentation for the ebBP and Signal Schemas

Due to size restrictions, the schema documentation for the ebBP and signal schemas are found in separate artifact files enclosed the ebBP v2.0.4 packages.