# OAGIS Implementation Using the ebXML CPP, CPA and BPSS specifications v1.0

# Table of Contents

**Introduction** **5**

*The Open Applications Group* *5*

*ebXML* *5*

*Rationale* *7*

*Scope* *8*

*Requirements* *8*

**Overall Approach** **9**

*ebXML Support for Third-Party Content* *9*

*Background* *10*
High-Level Comparison of ebXML and OAGI Specification Elements 10
Why Collaboration Definitions Are Important? 11

**Specifying ebXML Business Collaboration from OAGI Scenario Diagrams** **12**

*ebXML Semantics* *12*

*B2B Collaboration Definitions* *15*
Business Transaction Definitions 16
Document Definitions 19
Business Collaboration Choreography 21
Condition Expressions 27
Business Collaboration Failures 27
ebXML Signals 32
Patterns 36
Multiparty Collaborations 36

*Application-to-Application Collaboration Definitions* *38*
The ebXML Metamodel Subset 38
DTD 41
Example 46

**ebXML Collaboration Protocol Profile and Agreement** **48**

*Collaboration Protocol Profile* *48*

*Collaboration Protocol Agreement* *51*

**Bringing it All Together** **52**

*The Role of the OAGI Organization in Specifying CPPs and CPAs* *52*

*ebXML Functional Phases* *53*

**Appendix 1: ebXML Deliverables (www.ebXML.org)** **55**
Technical Specifications 55
Technical Reports 57
Reference Materials 58
White Papers 58

**Appendix 2: OAGI Scenario 55 RFQ / Quote** **59**

## Copyright Statements

This document represents an instance of utilization of the ebXML specifications and also describes a portion of the Open Applications Group specifications. There is no explicit or implied attempt to replace any part of these specifications. For the purposes of illustration, the author has copied sections of the ebXML specification or the OAGIS specification and has tried to make it explicit. The purpose of this document is to assist ebXML implementations with the Open Applications Group Integration Specification.

Special thanks to Connie Phillips, eXcelon, who spent long hours reviewing this document.

# Introduction

The OAGI (Open Applications Group, Inc.) has developed the largest set of business messages and integration scenarios for enterprise application integration and business-to-business (B2B) integration. However, OAGI does not specify an implementation architecture (also called an implementation framework). Three major B2B implementation architectures have been developed to this day: RosettaNet, BizTalk, and ebXML. They provide the basis for interoperability between different vendor and home-grown solutions alike. The OAGI policy is to be technology sensitive but not specific, and to use existing open standards when possible.

This white paper provides a detailed recommendation for how to transport OAGI BODs (Business Object Documents) (Version 7.1 dated April 25, 2001) using ebXML v1.0 dated 5/14/2001.

The reader is expected to have a thorough understanding of the OAGI BOD structure and the ebXML v1.0 specifications. The reader is encouraged to read the OAGI BOD Architecture documents (Section 1, Chapters 1 and 2) and the ebXML specifications. These documents can be freely obtained from the OAGI and ebXML Web sites.

## The Open Applications Group

The OAGI Integration Specification (OAGIS) includes a broad set of BODs and integration scenarios that can be used in different business environments, such as A2A and B2B. BODs are message definitions that can be used broadly across many different industries (for example, telecommunications and automotive) and aspects of Supply Chain Automation (for example, Ordering, Catalog Exchange, Quotes, etc.). OAGI also defines the OAMAS (Open Application Middleware API Specification), which is an application programming interface (API) for application integration that provides an abstraction from specific vendor solutions.

## ebXML

ebXML is a set of specifications that together enable a modular electronic business framework. ebXML enables a global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML-based messages. ebXML is jointly sponsored by the United Nations (UN/CEFACT) and the Organization for Structured Information Standards (OASIS).

There are four categories of ebXML deliverables:

- Technical Specifications
- Technical Reports
- Reference Materials
- White Papers

Technical Specifications are documents whose material fulfils the requirements of the ebXML Requirements document.

Technical Reports are documents that are either of the following:

- Guidelines: documents containing information to guide in the interpretation or implementation of ebXML concepts.
- Catalogs: documents containing foundation material based on ebXML Technical Specifications or Reports.

## Rationale

OAGI strives to adopt existing technology where possible and will benefit from the use of ebXML as an "implementation architecture." ebXML is one of the most advanced B2B implementation architecture available today. Its Messaging Service specifies how messages are communicated over a particular transport medium (for example, HTTP) and enables infrastructure-level interoperability among different vendor solutions. It also enables the specification of Collaboration Protocol Profiles (CPPs), which provide a machine-processable description of the capabilities of a particular company. This CPP can be published to an ebXML Registry and discovered by other Business Partners. Two CPPs can be composed to form a Collaboration Protocol Agreement (CPA) shared between two Business Partners, which specifies how these two companies have agreed to carry out a particular Business Collaboration.

The concept of a Business Collaboration is at the core of the design of ebXML and has been formalized as a "Business Process Definition," which is also known as a Collaboration Definition. ebXML is the first specification to provide the capability to model complete OAGI scenarios in a machine-processable Collaboration Definition. OAGI scenarios can be published as Collaboration Definitions that can then be used in CPPs and CPAs. These Collaboration Definitions could even be used with document formats that are not specified by OAGI, providing a new level of reuse for the work done by OAGI. Business Collaborations are made up of Business Transactions which themselves are composed of requesting and responding Business Actions. Like RosettaNet PIPs, ebXML transactions either succeed or fail, in which case, it would require each company to roll back its state to the one prior to the transaction. A Business Action is a message exchange between two partners and is analogous to a BOD.

Business Collaborations are central to the ability to do end-to-end collaborative commerce because they provide a way to express complete, complex, and legally binding "contracts" between Parties regardless of their respective application capabilities. In addition, a typical ebXML infrastructure would manage the state of each Collaboration instance, which does not require enterprise systems to understand or manage the complete end-to-end scenarios. Enterprise systems then can be organized into services that are bound to particular ebXML Business Actions of the Collaboration Definition. A Business Action is either a request or a response which compose an ebXML Business Transaction. Hence it is expected that, compared to RosettaNet, ebXML will enable far more complex ways of conducting electronic commerce while reducing the cost of implementation. By comparison, a RosettaNet infrastructure, or any service-based architecture, is merely a gateway between an application and Business Partners. On the other hand, an ebXML infrastructure provides the opportunity to integrate applications and Business Partners at the business process level.

In addition to the concept of Collaboration, ebXML introduces the notion of Registries. Registries enable Business Partners to publish the capabilities of their systems and applications, and to let other Business Partners discover this information in order to configure their own systems.

These series of specifications provide a robust framework in which to carry out sophisticated electronic commerce transactions. RosettaNet has made public its willingness to adopt the ebXML framework in lieu of RNIF 2.0. Microsoft has also announced some level of support.

Although ebXML provides an implementation architecture that can transport various content, a working group has been initiated to define “Core Components” which can be viewed as reusable pieces of content that can be assembled to create Business Documents such as invoices and purchase orders. This group has not yet completed its work and is not part of the voted specifications.

## Scope

This document provides a general recommendation for the usage and configuration of the ebXML implementation architecture in carrying out electronic transactions with OAGI BODs. In particular, it specifies how to define Collaboration Definitions for scenario diagrams. It also specifies how to declare CPPs and corresponding CPAs based on these Collaborations. The CPPs can be published to an ebXML Registry just like any other CPP. The Messaging Service can be used as is since it is content agnostic.

This document also describes the relationship between the infrastructure level messages (Signal Messages) used by ebXML and OAGI (Confirm BOD) to identify message receipt or exceptions.

This white paper does not deal with the aspects of ebXML that are not yet part of the specification (such as Core Components).

In addition, this paper recommends the use of Business Collaborations outside the context of ebXML. A Collaboration Definition can be used outside the scope of an ebXML solution and consequently could be used to formalized all OAGI scenario diagrams even the one that do not involve B2B communication. This approach may be desirable in order to provide a commercial OAGI solution with out-of-the-box capabilities while retaining the ability to customize the use of scenario diagrams for internal applications.

## Requirements

The primary requirements for the use of OAGI over ebXML:

- Use existing OAGI BODs without modification.
- Provide capability to exchange all OAGI BODs.
- Allow existing OAGI-based systems to operate with little or no change.

# Overall Approach

The tremendous value provided by ebXML, in addition to its robust Messaging Service, is in its ability to formalize all aspects of B2B relationships and express these formal specifications as machine-readable XML documents. This capability is a big advance in the realm of B2B since one of the drawbacks of present B2B solutions is the need to communicate a lot of design details between Parties and create specific or sometimes dedicated implementations to exchange Business Documents. RosettaNet had solved the communication problem by providing documents (PIP definitions) that each Party can implement without communicating; however, this translates into fairly rigid specifications that cannot be optimized and later discovered on a per partner basis. Furthermore, RosettaNet documents are not machine readable.

The approach we have taken in this paper is to specify a mapping between scenario diagrams and Business Collaborations in both a B2B and an A2A context. Then, we detail how CPPs and CPAs can be specified for the Business Collaborations. Even though, CPPs and CPAs are specific to each company a large portion can be specified at the Open Applications Group level, facilitating interoperability between OAGI implementations.

## ebXML Support for Third-Party Content

ebXML was designed to be content neutral. However, the ebXML Business Process Specification Schema (BPSS) imposes the following restrictions:

ebXML Signal Messages must be used as is; no third-party content can be included in these messages. Signal Messages are directly analogous to OAGI's ConfirmBOD and supplant its use within ebXML. However, to support existing OAGI-based applications that rely on ConfirmBOD, a relationship is defined such that a ConfirmBOD can be mapped to and from a Signal Message.

# Background

## High-Level Comparison of ebXML and OAGI Specification Elements

A brief comparison of the elements within the two specifications is provided in the following table. The shaded area indicates the scope of the current document.

Table 1. Specification-Level Comparison of RosettaNet and OAGI

| EbXML Specification Elements | OAGI Specification Elements |
|---|---|
| **Process Specification** covers integration scenarios that span multiple Binary Collaborations or Multiparty Collaborations. | **Scenario Diagrams** are abstract business system Collaboration diagrams that describe the possible interaction among business systems or components. |
| **Binary Collaboration** covers integration scenarios that span multiple Business Transactions. | No corresponding elements, even though some scenario diagrams may be as simple as a single Binary Collaboration. |
| **Multiparty Collaboration** covers integration scenarios that span multiple Binary Collaborations. These Collaborations can be assembled into a Multiparty Collaboration. | **Scenario Diagrams** are often specified between two roles and sometimes between more than two roles. Even in a two-role situation, scenario diagrams may contain more than one Binary Collaboration, therefore requiring the synthesis of multiple Binary Collaborations. |
| **Business Transaction** defines explicit Request and Response message exchange sequences (for example, Purchase Order Request – Purchase Order Acceptance). | **Sub-Scenarios** indicate possible Request and Response sequences (for example, ProcessPO – AcknowledgePO). |
| **Business Action** defines in particular the XML documents that are exchanged as part of the message. | **Business Object Document** (BOD) defines the XML message that is exchanged (for example, ProcessPO). |
| **Collaboration Protocol Profile (CPP)** specifies the capabilities of a given company. | No corresponding element |
| **Collaboration Protocol Agreement (CPA)** specifies the contract between two Business Partners as the intersection of two CPPs. | No corresponding element |
| **Registries** are global systems accessible by all via an API using the ebXML Messaging Service to publish and discover CPPs. | No corresponding element |
| **Messaging Service** defines the communication and transport mechanisms for exchanging messages. | No corresponding element |

OAGI defines integration scenario diagrams that suggest how BODs are exchanged. In B2B scenarios, this lack of rigidity is more a liability than an asset. One of the goals of this white paper is to provide the framework for systematically developing an ebXML process specification for each OAGI scenario diagram. Shortly, OAGI will publish an ebXML version of its scenario diagram and create the corresponding CPP templates. Full CPPs cannot be created because they need to contain information specific to each company. Alternatively, companies may create their own version of a Collaboration Definition based on an OAGI scenario diagram and publish on ebXML Registries for their partners to discover.

Certain BODs can contain response messages such as the ProcessPO and AcknowledgePO messages. However, most have an implied message exchange

sequence such as the Get/Show and GetList/List combinations of BODs. (for example, GetPO – ShowPO and GetListPO – ListPO).

**Why Collaboration Definitions Are Important?**

Business collaboration is a fairly new concept in the software industry. As part of a Collaboration Protocol Agreement (CPA), it represents a "contract" between two or more Business Partners which explicitly defines when and why a specific message should be sent or received. This part of the "contract" provides a shared understanding of the interaction. The way the contract is implemented is private but the expectations are public. In an environment where your organization may receive tens of thousands of business messages per day, it is important to be able to relate a context to each and every one of them. This is the goal of the BPSS specification. The infrastructure that enforces Collaborations acts as a business firewall, allowing messages that initiate a Collaboration and subsequently only the messages that precisely follow the sequencing rules of Business Transaction Activities and their respective transitions.

The closest analogy to a Collaboration is two APIs that need to invoke each other. Most often, a system will publish an API that other systems can invoke without necessarily exposing their functionality and capabilities to the original system. In the B2B world, both systems are most often stateful and we need to specify the interaction with great detail to be able to configure each system. At this point, the API (or services) is almost irrelevant because the goal of the system is to complete the Collaboration, not necessarily a function call.

Collaboration Definitions are particularly important in two cases: a large company that may have multiple system performing the same function (for example, procurement systems) and small-to-medium Business Partners that need to interact with diverse groups of partners.

In the first case, a Collaboration allows the company to expose a common business logic regardless of the Business Rules specific to each application. It also provides an opportunity to "outsource" some of the validation outside the application, leaving the application focusing on its business problem (processing an invoice) rather than deciding if this invoice is coming from an authorized source and is part of a valid on-going transaction. This approach to developing new business systems will ensure a greater overall maintainability. ebXML provides a machine-readable way of specifying the business logic, which allows us to move from a procedural approach to a declarative approach.

A Business Partner dealing with various groups, each of which supports slightly varying Collaborations, is of equal interest. An ebXML infrastructure will enforce each agreement and allow the company to reconcile each one of them with a common processing system (such as order entry).

ebXML Collaboration Definitions open up a new way to carry out commerce activity that resemble the way business is done today with phone, fax mail, and email while enabling a new application programming model in which a lot of the "business process" logic is outsourced, and not hard-coded.

# Specifying ebXML Business Collaboration from OAGI Scenario Diagrams

The goal of the ebXML Business Process Specification Schema is to support the specification of Business Transactions and the choreography of Business Transactions into Business Collaborations. Each Business Transaction can be implemented using one of many available standard UN/CEFACT Modeling Methodology (UMM) patterns. These patterns determine the actual exchange of Business Documents and business signals between the partners to achieve the required electronic commerce transaction. Business Transactions that do not follow UMM patterns can also be defined.

OAGI uses scenario diagrams for both the specification of A2A and B2B BOD interchanges. Sometimes scenarios can actually be implemented in both an A2A (system-to-system) or B2B (company-to-company) configuration. In the following two sections, we cover the use of Collaboration Definitions for both B2B and A2A scenarios. We will use examples to illustrate our approach: Scenario 55.0 BUYER AND SUPPLIER RFQ - QUOTE SCENARIO, as well as Scenario 1.0 GENERAL LEDGER TO SUB-LEDGER SCENARIO. The complete description of these scenarios can be found on the S2_Scenarios.doc document of the OAGI specification.

## ebXML Semantics

A Business Collaboration is essentially the specification of Business Transaction Activities between two roles, their associated document flow, and the choreography of these Business Transaction Activities. It is important to note that the sequencing rules contained in a Collaboration Definition are not between messages but between Business Transaction Activities. It is also important to note that the specification distinguishes between Business Transactions and Business Transaction Activities. A Business Transaction can be viewed as a type declaration, while Business Transaction Activities (which reference a unique business transaction type) are the usage of this transaction within a particular choreography. In particular, several references to the same Business Transaction may appear several times in the same Collaboration Definition. And a Business Transaction may sometimes be used either way between the two roles.

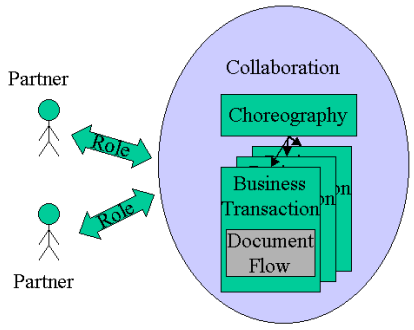


Figure 1. Basic Semantics of a Business Collaboration[1]

Reading the entire ebXML Business Process Specification Schema document is highly recommended. Here is an extract of the Business Transaction definition:

A Business Transaction[1] is the atomic unit of work in a trading arrangement between two business partners. A Business Transaction is conducted between two parties playing opposite roles in the transaction. The roles are always a requesting role and a responding role.

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

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

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

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

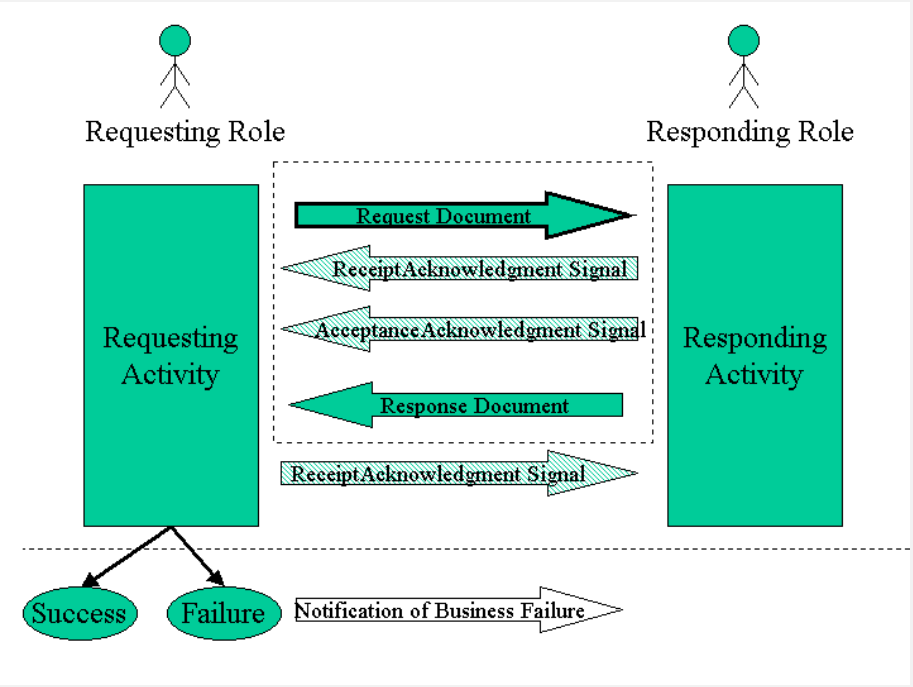



Figure 2. Schematic of Core Business Transaction Semantics

Scenario diagrams needs to be organized as Business Transactions, also called sub-scenarios in the OAGI specification.

Some semantics attached to the Business Transaction or the Business Collaboration definitions are specific to transactions between companies. For instance, the Business Actions that form a Business Transaction can be specified with the *isNonRepudiationRequired* or *IsGuaranteedDelivery* flag set to true. On the other hand, parameters such as the *timeToPerform* attribute of a Business Transaction can be suited for both situations – B2B and A2A.

## B2B Collaboration Definitions

In this section, we illustrate our approach with Scenario 55. Each Business Transaction needs to be specified and choreographed. This choice is somewhat arbitrary. In addition, the usage of ConfirmBOD is not necessarily required as the semantics of a Business Transaction allow for both a receipt and a business acknowledgement prior to sending the response to a request.

This example is a Multiparty Collaboration (a buyer, a seller, and an intermediary such as a marketplace), composed of two Binary Collaborations.

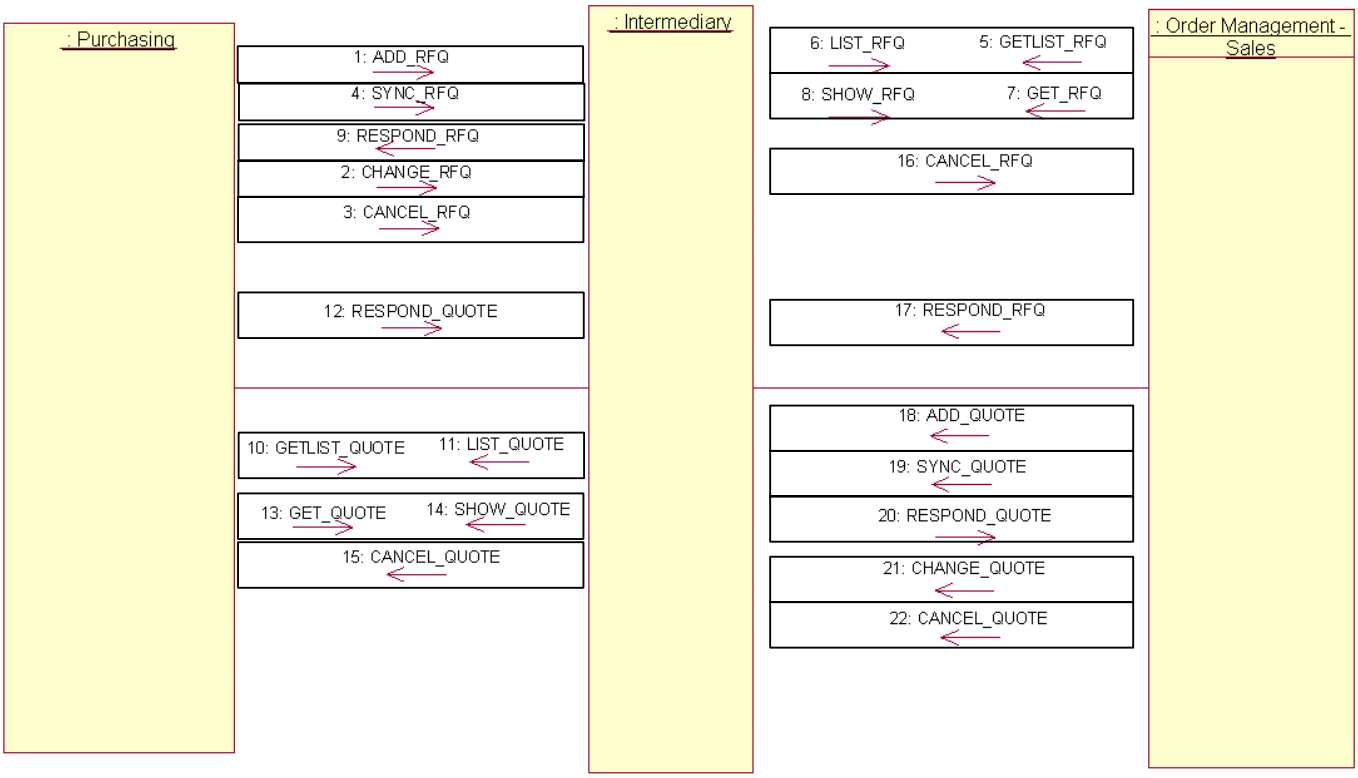



Figure 3. OAGI Scenario 55 divided as ebXML Business Transactions

### Business Transaction Definitions

The Business Transaction is a key concept in ebXML BPSS. As mentioned earlier, the semantics are extremely precise and were designed to carry out reliably complex commerce activities between Business Partners.

OAGI does not have a corresponding element. One of the challenges in defining Business Transactions is to package BODs as requests and responses. The request is usually easy to identify; however, the response is trickier. The ebXML semantics specify that a responding activity's document envelope may contain multiple attachments. A BOD corresponds to one attachment. In addition, a responding activity may have multiple possible response document envelopes, each containing any number of attachments (Figure 4), out of which one and only one document envelope will be sent as the response. The *isPositiveResponse* attribute indicates if the response is meant to be positive or negative; this parameter is associated with the "business failure" of the business transaction (see the section titled "Business Collaboration Failures").

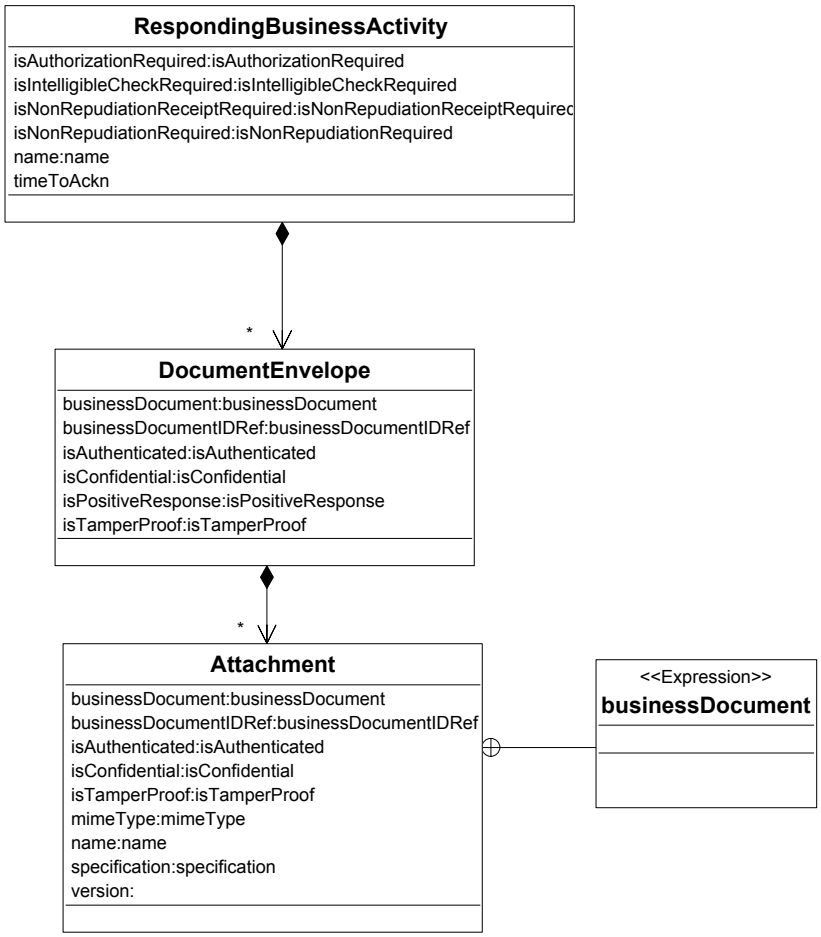



Figure 4. ebXML Document Envelope

Transitions from the Business Transaction can be based on the success or failure of the transaction, as well as by guard expressions on the content of the document if the success cannot be determined by the document envelope alone.

A possible representation of a Business Transaction is to use UML activity diagrams:

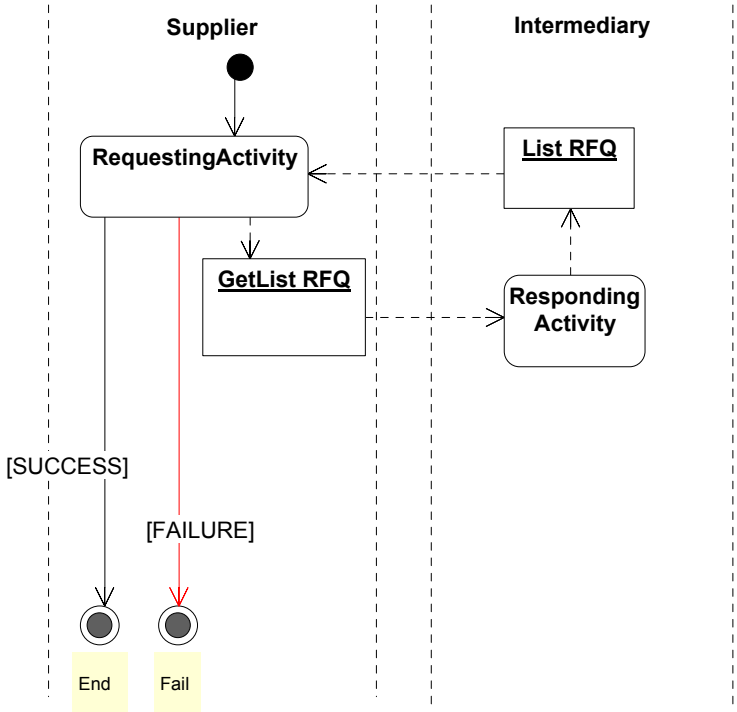



Figure 5. GetList RFQ Business Transaction

The UML notation represents the two roles as swim lanes, the exchange of documents as object flows, and the start and end of the Business Transaction. This notation is in no way part of the specification. However, it is being used by the RosettaNet and UN/CEFACT as part of the UMM effort, of which ebXML BPSS is a semantics subset.

MEGA Int. offers an alternative but proprietary representation:

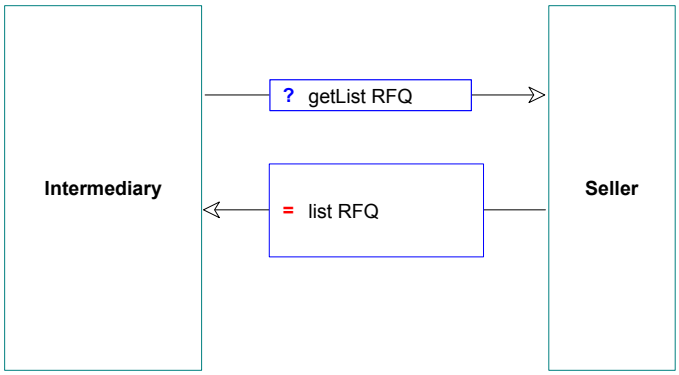



Figure 6. GetList RFQ Business Transaction with MEGA's Notation

Here is the corresponding Business Transaction definition:

```
<BusinessTransaction name=" BT:Get RFQ List"
isGuaranteedDeliveryRequired="true">
   <RequestingBusinessActivity isNonRepudiationRequired="true"
                               timeToAcknowledgeReceipt="PT1M"
                               timeToAcknowledgeAcceptance=""
                               isAuthorizationRequired="true"
                               isIntelligibleCheckRequired="true"
                               isNonRepudiationReceiptRequired="false"
                               isNonRepudiationRequired="false">
      <DocumentEnvelope businessDocument="Getlit RFQ"
                        isPositiveResponse="false"
                        isAuthenticated="false"
                        isConfidential="false"
                        isTamperProof="false" />
  </RequestingBusinessActivity>
  <RespondingBusinessActivity isAuthorizationRequired="false"
                              isIntelligibleCheckRequired="true"
                              isNonRepudiationReceiptRequired="false"
                              isNonRepudiationRequired="true">
     <DocumentEnvelope businessDocument="Showlist RFQ"
                       isPositiveResponse="true"
                       isAuthenticated="false"
                       isConfidential="false"
                       isTamperProof="false" />
  </RespondingBusinessActivity>
 </BusinessTransaction>
```

All the attributes above are related to the “quality of service” required when performing the Business Transaction. The *timeToPerform* attribute is actually defined late when the Business Transaction is used in the context of a Binary Collaboration as a Business Transaction Activity.

We provide a few guidelines about using these attributes. First, in rare cases, someone would need to use an Acceptance Acknowledgement signal. A Receipt Acknowledgement will confirm that the request (or response) has been received, that it has been somehow archived, and that it is valid with respect to its schema definition. An Acceptance Acknowledgement would add that internal Business Rules about the request have been validated. This is often redundant with the response message itself. So we recommend in most cases to use the receipt signal only.

Other parameters are expensive to implement or deploy and should be used appropriately:

1) Non-Repudiation and Legally Binding often require the storage of all the messages and receipts for extended periods of time (months or years).
2) Confidentiality and Tamper Proof documents require digital signature and encryption technologies.

Lastly, we recommend to set the timeout parameters in a way that does not generate a lot of unnecessary exceptions. A typical mistake is to set their value to the expected value rather than the maximum value.

### Document Definitions

Business Documents that participate in Business Transactions are defined by their name, specification element, and specification location. The specification element is a reference to the element within the schema definition that defines this document.

***DocumentEnvelope*** references a primary document as indicated in the following figure. Several attachments can be added to a document envelope, each of which references a Business Document as well. Attributes such as *isConfidential*, *isAuthenticated*, and *isTamperProof* deal with the security of the document between parties as well as within a party of the collaboration.

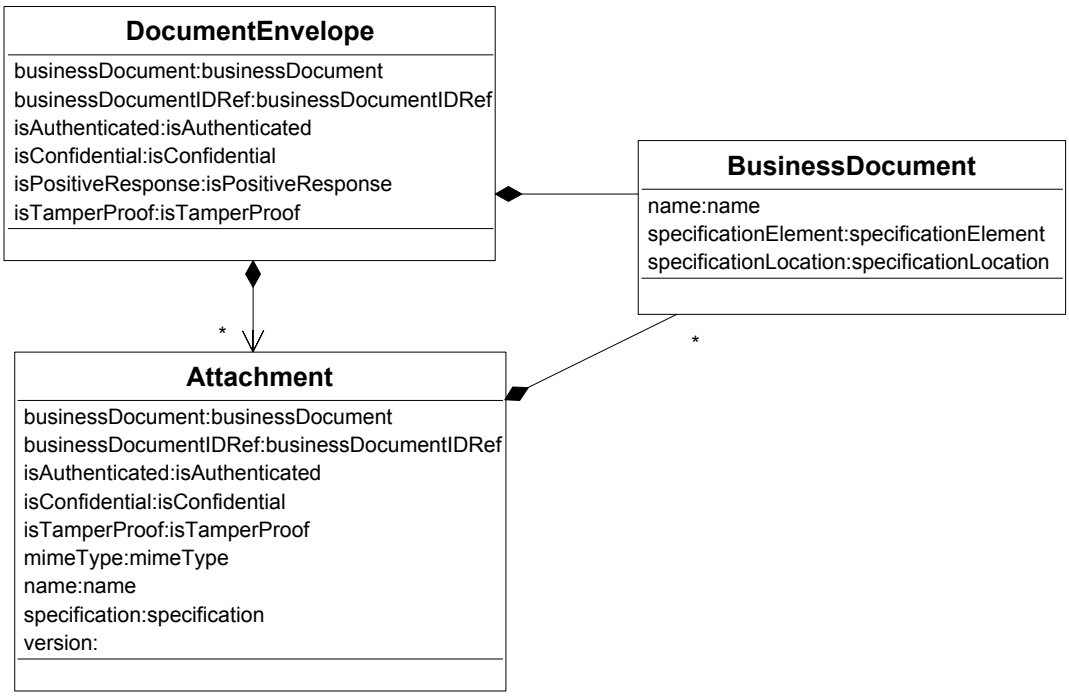



Figure 7. ebXML Business Document

For example, the following ***BusinessDocument*** definition:

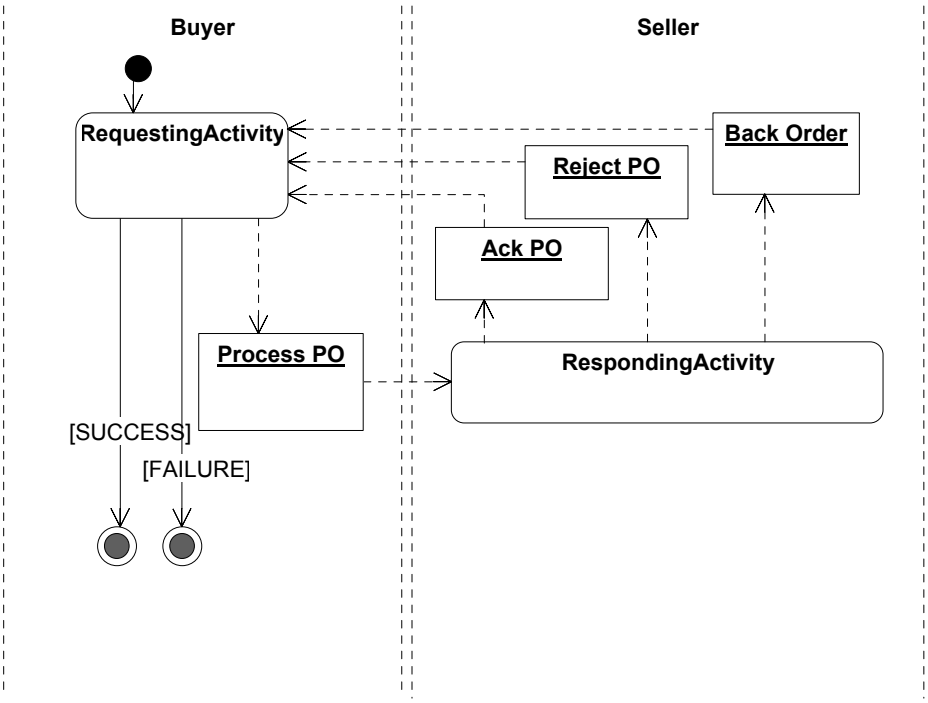
```
<BusinessDocument name="Getlist RFQ"
                  specificationLocation=
          ”www.openapplications.org/OAGIS/v7.1/148 getlist rfq 003.dtd”
                  specificationElement=”GETLIST_RFQ_003” />
```

can be used in a ***DocumentEnvelope*** definition as:

```
<RequestingBusinessActivity name="Request RFQ List">
       <DocumentEnvelope businessDocument="Getlist RFQ"/>
</RequestingBusinessActivity>
```

Condition expressions can be attached to a document to indicate the intent of a particular response. For instance, in a Process Purchase Order transaction, there can be three possible “logical” responses that are all carried via the same BOD (Acknowledge Purchase Order).

Figure 8. Process Purchase Order Business Transaction with Three Possible Responses




```
<BusinessDocument name="Ack PO"
specificationLocation="http://www.openapplications.org/OAGIS/v7.1/004 acknowl
edge_po_007.dtd" specificationElement="ACKNOWLEDGE_PO_007">
  <ConditionExpression expressionLanguage="XPath"
expression="//STATUSLVL="00"" />
</BusinessDocument>
<BusinessDocument name="Reject PO"
specificationLocation="http://www.openapplications.org/OAGIS/v7.1/004 acknowl
edge_po_007.dtd" specificationElement="ACKNOWLEDGE_PO_007">
  <ConditionExpression expressionLanguage="XPath"
expression="//STATUSLVL="01"" />
</BusinessDocument>
<BusinessDocument name="Back Order"
specificationLocation="http://www.openapplications.org/OAGIS/v7.1/004 acknowl
edge_po_007.dtd" specificationElement="ACKNOWLEDGE_PO_007">
  <ConditionExpression expressionLanguage="XPath"
expression="//STATUSLVL="02"" />
</BusinessDocument>
```

These “logical” Business Documents can be used to express sequencing rules in a Binary Collaboration Definition. One advantage of this approach, rather than making the condition explicit in the Collaboration Definition, is the isolation of the document format and the rule that depends on it such that new versions of a format (or different formats altogether) can be used without impacting the logic of the Collaboration Definition itself, provided the intent of the “logical” document is respected.

### Business Collaboration Choreography

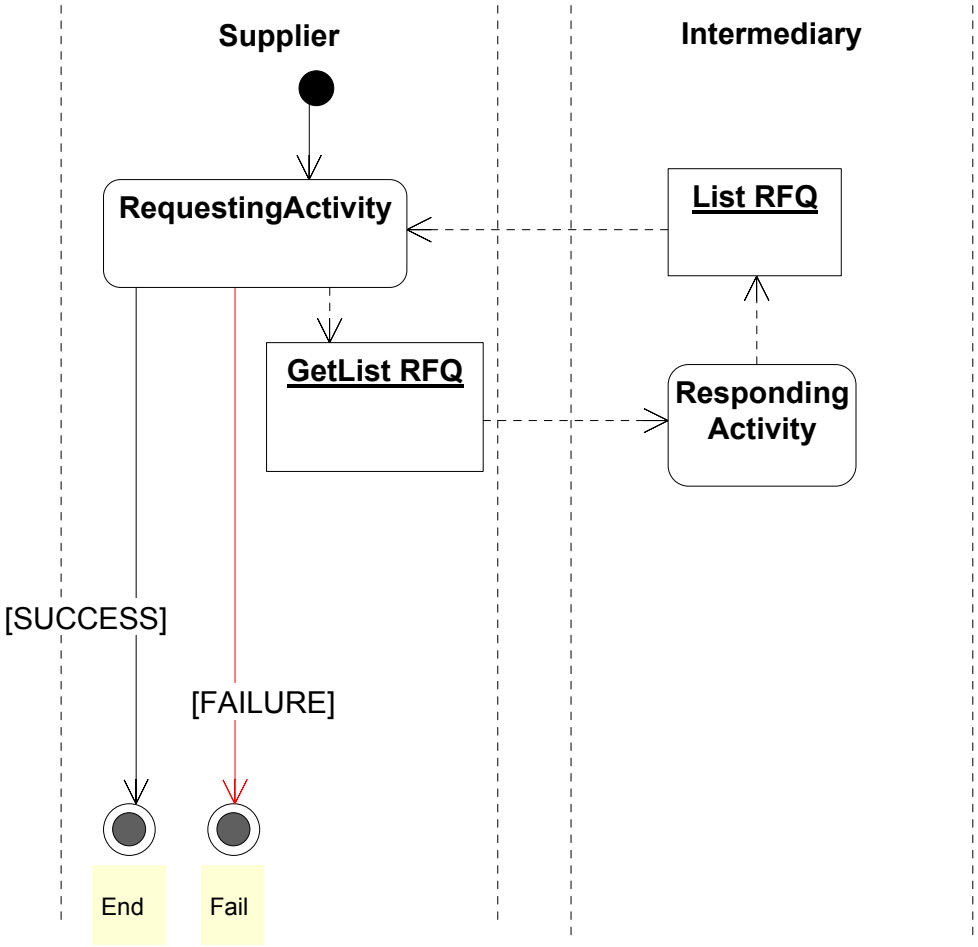
Once all Business Transactions have been defined, we can start to specify the choreography of the Collaboration. As UML profiles, Collaborations can be represented with UML artifacts. The UML notation does not readily provide the ability to represent ebXML Collaborations in their entirety. For instance, “UML Collaboration diagrams” cannot represent sequencing rules. This section follows the UN/CEFACT Modeling Methodology (UMM) guidelines. However, in the present state, such representation will not be able to capture enough details in order to have an isomorphic relationship with the XML definition. We recommend to always refer to the XML document for the specification, and use only UML artifacts as a guide in understanding the content of the XML document, which may sometimes become very large.

We chose to represent a Collaboration in two views:

- Business Transactions: activity diagrams
- Binary Collaboration: activity diagrams

The following figure represents the getList RFQ Business Transaction definition.

Figure 9. getList RFQ Business Transaction Representation with a UML Activity Diagram



A Collaboration and its sequencing rules are represented with an activity diagram:

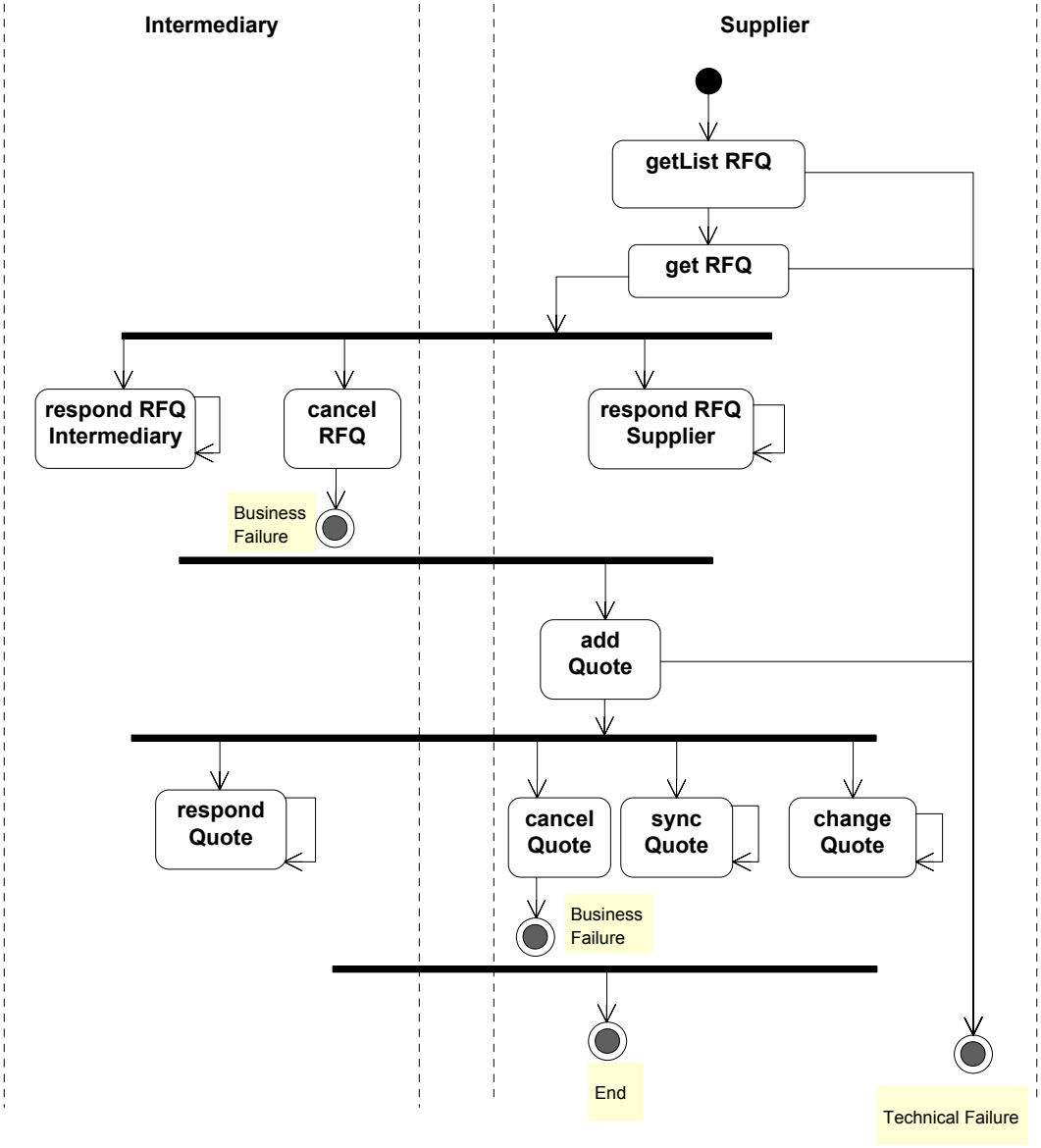



Figure 10. Collaboration Activity Diagram

An activity within a swim lane indicates that this Business Transaction Activity is initiated by the corresponding role. Consequently, this notation is not appropriate for representing multiparty Collaborations since the receiving Party is implied rather than explicit. This is not the only shortcoming of the notation. Negotiation patterns such as the ones above are hard to represent without breaking the rules of an activity diagram.

A Business Transaction Activity that can be repeated indefinitely, such as the respond RFQ (questions and answers about the RFQ), is represented with a transition to itself:

In particular, we mean that in the event of a success or a failure of this transaction, the supplier is allowed to send further questions about the RFQ. In this case, there is no transition out of this Business Transaction Activity because the way the collaboration proceeds is via a timeout.

However, the way the ebXML specification was designed, one cannot specify a timeout for the Fork/Join elements. The way to design this Collaboration in a way where one can specify the proper timeouts is to aggregate the corresponding Business Transaction Activities into a Collaboration Definition. This Collaboration Definition can then be used as a Collaboration activity as represented in the following figure.

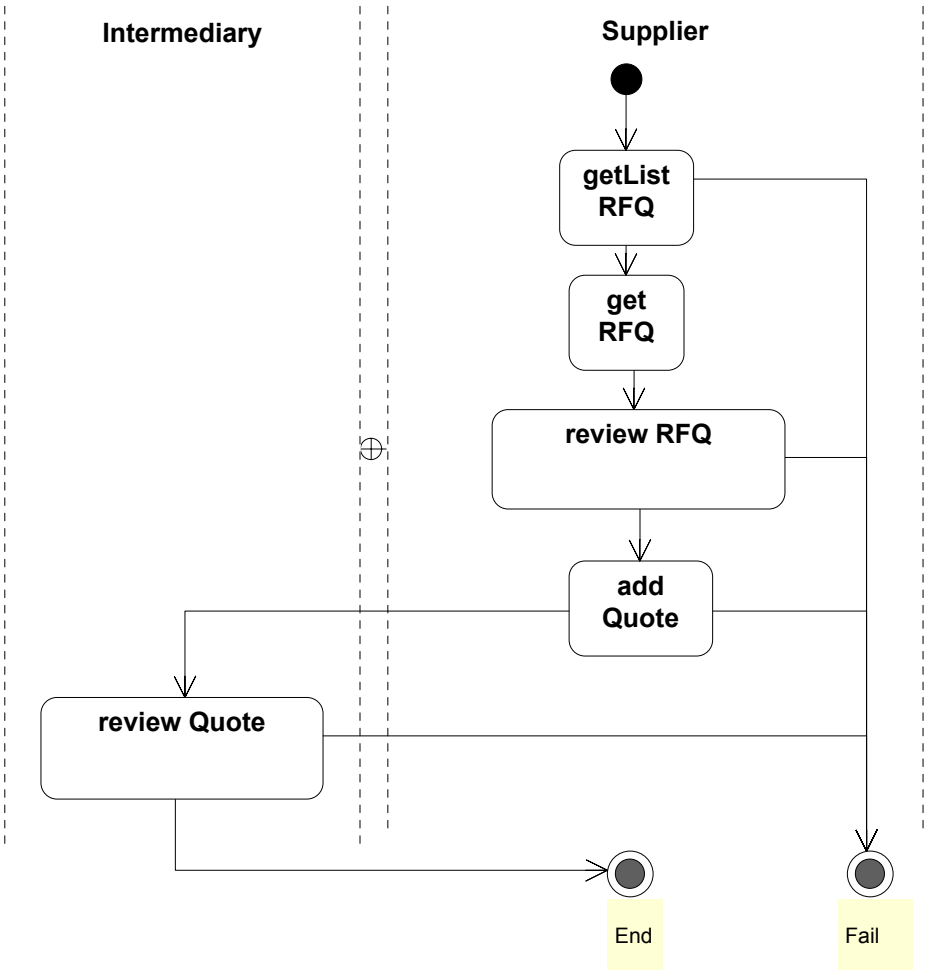
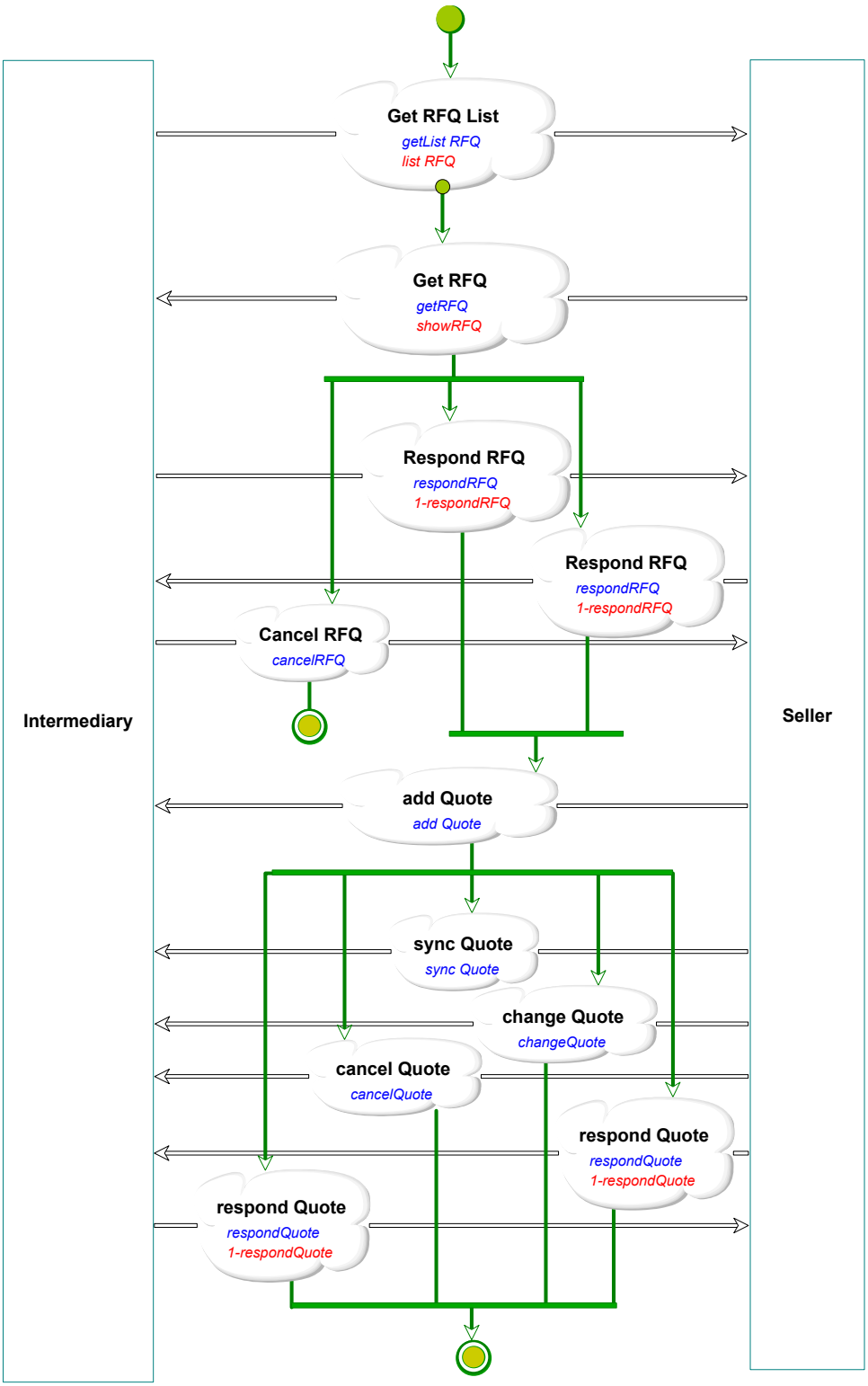



Figure 11. A Collaboration Definition Used as a Collaboration Activity

MEGA Int. offers a proprietary notation, which we show here:

Figure 12. MEGA Notation



Binary Collaboration definitions refer to Business Transaction Activity definitions (which are a usage of a Business Transaction definition) and represent the states of the activity diagrams. The transitions are usual activity diagram transitions.

```
 <BinaryCollaboration name="oagi:55.0 INTERMEDIARY AND SUPPLIER RFQ - QUOTE
SCENARIO " timeToPerform="P30D">
        <Documentation>timeToPerform = Period: 30 days from start of
transaction</Documentation>
        <InitiatingRole name="supplier"/>
        <RespondingRole name="intermediary"/>
        <BusinessTransactionActivity name="Get RFQ List"
               businessTransaction="BT:Get RFQ List"
               fromAuthorizedRole="supplier"
               toAuthorizedRole="intermediary"/>
        <BusinessTransactionActivity name="Get RFQ"
               businessTransaction="BT:Get RFQ"
               fromAuthorizedRole="supplier "
               toAuthorizedRole="intermediary "/>
        <Start toBusinessState="Get RFQ List"/>
        <Transition fromBusinessState="Get RFQ List"
                    toBusinessState="Get RFQ"/>
        ...
 </BinaryCollaboration>
```

The next part of the Collaboration uses a Fork element to specify that multiple Business Transaction Activities can happen in parallel. In this phase, there are actually two Business Transaction Activities that share the same Business Transaction definition: Respond RFQ. This activity can either be initiated from the intermediary role or from the supplier role.

```
<BinaryCollaboration name="oagi:55.0A REVIEW RFQ" timeToPerform="P30D">
        <InitiatingRole name="intermediary"/>
        <RespondingRole name="supplier"/>
        <BusinessTransactionActivity name="Cancel RFQ"
               businessTransaction="BT:Cancel RFQ"
               fromAuthorizedRole="intermediary"
               toAuthorizedRole=""/>
        <BusinessTransactionActivity name="Respond RFQ Intermediary"
               businessTransaction="BT:Respond RFQ"
               fromAuthorizedRole="intermediary"
               toAuthorizedRole=""/>
        <BusinessTransactionActivity name="Respond RFQ Supplier"
               businessTransaction="BT:Respond RFQ"
               fromAuthorizedRole="intermediary"
               toAuthorizedRole=""/>
        <Fork name="Review RFQ Start" nameId="Review RFQ Start"/>
        <Start toBusinessState="Review RFQ Start"/>
        <Transition fromBusinessState="Review RFQ Start"
                    toBusinessState="Cancel RFQ"/>
        <Transition fromBusinessState="Review RFQ Start "
                    toBusinessState="Respond RFQ Intermediary "/>
        <Transition fromBusinessState="Review RFQ Start "
                    toBusinessState="Respond RFQ Supplier"/>
```

```
        <Join name="Respond RFQ End" nameid="Review RFQ End"
              waitForAll="No"/>
        <Failure fromBusinessState="Cancel RFQ"/>
        <Transition fromBusinessState="Respond RFQ Intermediary"
                    toBusinessState="Respond RFQ Intermediary"/>
        <Transition fromBusinessState="Respond RFQ Supplier"
                    toBusinessState="Respond RFQ Intermediary"/>
</BinaryCollaboration>
```

This part of the Collaboration Definition uses the "Failure" element, which indicates that the Collaboration ends if the Cancel RFQ Business Transaction completes "successfully".

There is, however, an issue with the Collaboration Definition, specifically with the way it was defined as part of Scenario 55. All the BODs that are exchanged as part of the last phase of the Collaboration leave the Collaboration open ended, meaning that one could wait an indefinite amount of time to expect a Change Quote BOD. Some people who have actually implemented this Collaboration are sending an email to the suppliers as a notification of acceptance of the quote. Even though the ebXML BPSS provides the tools to describe such an event in a Collaboration Definition, it might be more appropriate to specify a BOD, as mundane as the ConfirmBOD, to end the Collaboration unambiguously.

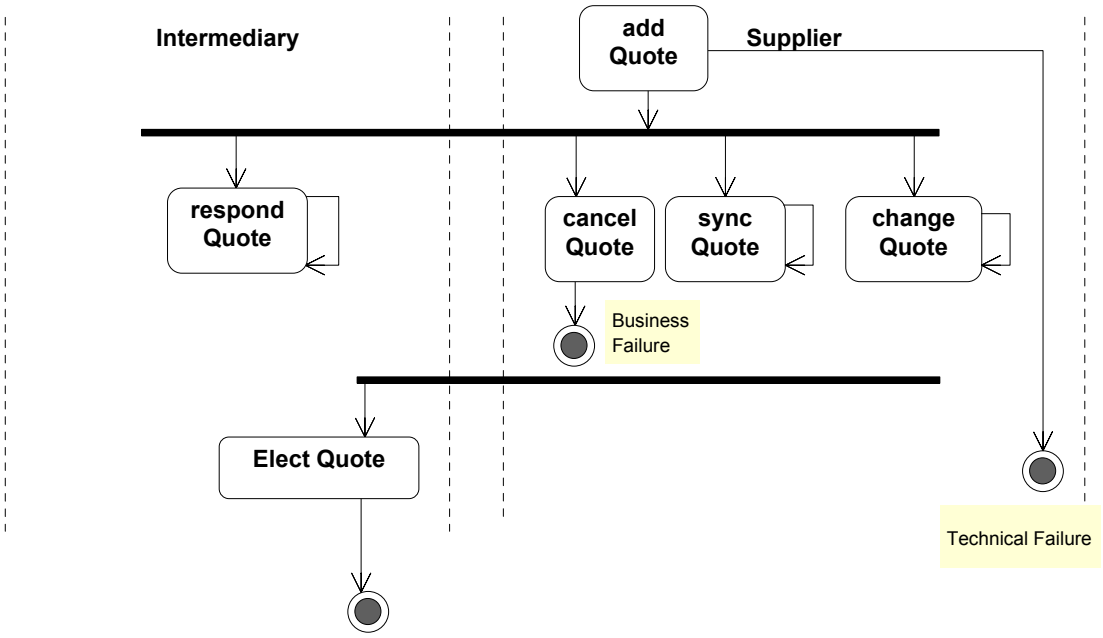



Figure 103. Unambiguous End for the Collaboration Definition Using the Elect Quote Business Transaction

### Condition Expressions

Guards can be defined for any transitions. There are two ways to specify guards.

1) A Business Transaction definition may establish clearly specific end states such as Success, AnyFailure, BusinessFailure, and TechnicalFailure. Transitions from state to state or to completion state can be based on this attribute value. This presents some advantages over method number 2 below, since having an explicit expression as a guard might tie the Collaboration Definition to document formats, for instance, which may not be desirable since formats can evolve over time. It might also prevent the use of the Collaboration with document formats other than BODs.

   So typically, each transition should either have present a *guardCondition* attribute or a ***ConditionExpression*** (see paragraph 2) below).

```
<Transition fromBusinessState="Get RFQ"
            toBusinessState="Respond RFQ Start"
            guardCondition=”Success”/>
```

   This means that in Scenario 55, one can only start the Respond RFQ activity if one successfully completed the transaction Get RFQ. It is critical for the well being of every ebXML application that these guards are in place to explicitly end the Collaboration when a failure occurs.

2) ***ConditionExpressions*** can be explicit statements, for instance, in Java or as XPath predicates. This example shows how the guard can be defined to identify whether there is actually an RFQ in the response document envelope.

```
<Transition fromBusinessState="Get RFQ"
            toBusinessState="Respond RFQ Start">
        <ConditionExpression expressionLanguage=”XPATH”
                             expression=”//RFQ”>
</Transition>
```

### Business Collaboration Failures[1]

The following subsections discuss the two causes of failure[1]: Timeouts and Exceptions. When either one happens, it is the responsibility of the two roles to do the necessary roll-back, and to exit the transaction. The responsibilities of the two roles differ slightly and are described in each of the sections below. Generally, if a failure happens at the responding role, the responding role will send an exception signal to the requesting role, and both Parties will exit the current transaction. If a failure happens at the requesting role, the requesting role will exit the current transaction and in a separate transaction notify the responding role about the failure. This way the flow of control within a transaction is always unambiguous and finite.

[1] Note that this section is mostly a copy of the corresponding section in the ebXML BPSS document.

## Business Failures

Business failures are solely related to the agreed upon intend of the response document envelope marked by the *isPositiveResponse* property. The condition expressions on the content of a document that lead to a business transaction business failure are defined at the business document level. These “logical documents” can be related to a clear intent on the success or failure of the business transaction they are part of.

Consequently, the *BusinessFailure* on a ***Transition*** *conditionGuard* is true when the response document envelope is marked with *isPositiveResponse*=false.

## Technical Failures

There are two types of technical failures: timeouts and exceptions.

### Timeouts

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

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

| Response Required | Parameter Name | Meaning of Timeout |
|---|---|---|
| Receipt acknowledgement | *timeToAcknowledgeReceipt* | The time a responding role has to acknowledge receipt of a Business Document. |
| Acceptance Acknowledgement (Non-substantive) | *timeToAcknowledgeAcceptance* | The time a responding role has to non-substantively acknowledge business acceptance of a Business Document. |
| Substantive Response | *timeToPerform* | The time a responding role has to substantively acknowledge business acceptance of a Business Document. |

A timeout parameter must be specified whenever a requesting partner expects one or more responses to a Business Document request. A requesting partner must not remain in an infinite wait state.

The timeout value for each of the timeout parameters is absolute, that is, not relative to each other. All timers start when the initial requesting Business Document is sent. The timer values must comply with the well-formedness rules for timer values.

A responding partner simply terminates if a timeout is thrown. This prevents responding Business Transactions from hanging indefinitely. A requesting partner terminates if a

timeout is thrown and then sends a notification of failure to the responder as part of a separate transaction.

When the time to perform an activity equals the time to acknowledge a receipt or the time to acknowledge business acceptance, then the highest priority timeout exception must be used when the originator provides a reason for revoking the original Business Document offer. The *timeToPerform* exception is lower priority than both the *timeToAcknowledgeReceipt* and the *timeToAcknowledgeBusinessAcceptance*.

Two parameters are defined in the Business Transaction definition:

```
<BusinessTransaction name="BT:Add Quote">
 <RequestingBusinessActivity name="Request RFQ List"

        timeToAcknowledgeAcceptance=”P24H”
        timeToAcknowledgeReceipt=”P1H”>

        <DocumentEnvelope businessDocument="Add Quote"/>
 </RequestingBusinessActivity>
</BusinessTransaction>
```

While the time to perform a Business Transaction is defined at the Business Transaction Activity level (that is, it can be specific to each instance of the Business Transaction):

```
        <BusinessTransactionActivity name="Respond RFQ Supplier"
              businessTransaction="BT:Respond RFQ"
              fromAuthorizedRole="intermediary"
              toAuthorizedRole=""

              timeToPerform=”P2D”/>
```

One may want to be optimistic and not put too many constraints on timeout. In particular, *timeToAcknowledgeReceipt* should be larger than a typical downtime of an ebXML system; otherwise, a lot of timeouts might be generated without a particular purpose.

Binary Collaborations also have a *timeToPerform* attribute. This is particularly useful when a timeout needs to be associated to a group of Business Transactions rather than a single transaction. In the RFQ/Quote example, there are two open-ended periods during which one may want to enable Business Transactions but without necessarily expecting them to occur. For instance, the respond RFQ, respond Quote, and Cancel Quote are all possible Business Transactions, but relatively unlikely in a regular scenario. The only way to express a global timeout to specify that the periods of reviewing the RFQ or the quote are completed is to specify these Business Transaction Activities as part of a “sub-collaboration” and define a Collaboration activity within the main Collaboration Definition, which is an instance of this Collaboration activity.

```
 <BinaryCollaboration name="oagi:55.0B QUOTE REVIEW" timeToPerform="P5D">

...
</BinaryCollaboration>

 <BinaryCollaboration name="oagi:55.0A RFQ REVIEW" timeToPerform="P10D">
```

```
...
</BinaryCollaboration>
 <BinaryCollaboration name="oagi:55.0 INTERMEDIARY AND SUPPLIER RFQ - QUOTE
SCENARIO " timeToPerform="P30D">
         ...
         <CollaboratioActivity name="Review RFQ"
                binaryCollaboration=" oagi:55.0 INTERMEDIARY AND SUPPLIER RFQ
- QUOTE SCENARIO:RFQ REVIEW "
                fromAuthorizedRole="intermediary"
                toAuthorizedRole=""/>

         <BusinessTransactionActivity name="Add Quote"
                businessTransaction="BT:Add Quote"
                fromAuthorizedRole="supplier"
                toAuthorizedRole="intermediary"/>

         <Transition fromBusinessState="Review RFQ"
                     toBusinessState="Add Quote"/>

        <CollaboratioActivity name="Review Quote"
                binaryCollaboration=" oagi:55.0 INTERMEDIARY AND SUPPLIER RFQ
- QUOTE SCENARIO:QUOTE REVIEW "
                fromAuthorizedRole="intermediary"
                toAuthorizedRole=""/>

        <Transition fromBusinessState="Add Quote"
                     toBusinessState="Review Quote"/>

         ...

 </BinaryCollaboration>
```

### Exceptions

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

- Control Exception

  A ControlException signals an error condition in the management of a Business Transaction. This business signal is asynchronously returned to the initiating activity that originated the request. This exception must terminate the Business Transaction. **These errors deal with the mechanisms of message exchange** such as verification, validation, authentication, and authorization and will occur up to message acceptance. Typically, the rules and constraints applied to the message will have only dealt with structure, syntax, and message element values.

- Business Protocol Exceptions

A business protocol exception (or ProcessException) signals an error condition in a Business Activity. This business signal is asynchronously returned to the initiating role that originated the request. This exception must terminate the Business Transaction. **These errors deal with the mechanisms that process the business transaction** and will occur after message verification and validation. Typically, the rules and constraints applied to the message will deal with the semantics of message elements and the validity of the request itself. The content is not valid with respect to a responding role's Business Rules. This type of exception is usually generated after an AcceptanceAcknowledgement has been returned.

A business protocol exception terminates the Business Transaction. The following are business protocol exceptions:

- *Negative acknowledgement of receipt*. The structure/schema of a message is invalid.
- *Negative acknowledgement of acceptance*. The business rules are violated.
- Performance exceptions. The requested Business Action cannot be performed.
- *Sequence exceptions*. The order or type of a Business Document or business signal is incorrect.
- *Syntax exceptions*. There is invalid punctuation, vocabulary, or grammar in the Business Document or business signal.
- *Authorization exceptions*. Roles are not authorized to participate in the Business Transaction.
- *Business process control exceptions*. Business Documents are not signed for non-repudiation when required.

A Business Transaction is defined in very atomic and deterministic terms. It always is initiated by the requesting role, and will always conclude at the requesting role. Upon receipt of the required response and/or signals, or timeout of same, the requesting role can unambiguously determine the success or failure of the Business Transaction.

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

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

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

The consequence for someone writing a Collaboration Definition corresponding to and OAGI scenario is that all possible exceptions must be handled specifically. The parameters below specify the exceptions that will be trapped by the ebXML application such that they can specifically trigger the corresponding Business Transaction Activities that handle them.

**IsAuthorizationRequired**

If a partner role needs authorization to request a Business Action or to respond to a Business Action, then the sending partner role must sign the Business Document

exchanged and the receiving partner role must validate this business control and approve the authorizer. A responding partner must signal an authorization exception if the requesting partner role is not authorized to perform the Business Activity. A sending partner must send notification of failed authorization if a requesting partner is not authorized to perform the responding Business Activity.

**IsNonRepudiationRequired**

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

**isNonRepudiationOfReceiptRequired**

Both partners agree to mutually verify receipt of a requesting Business Document and that the receipt must be non-repudiatable. A requesting partner must send notification of failed business control (possibly revoking a contractual offer) if a responding partner has not properly delivered their Business Document. For a further discussion of non-repudiation of receipt, see also the ebXML E-Commerce and Simple Negotiation Patterns.

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

- Verify responding role identity (authenticate) – Verify the identity of the responding role (individual or organization) that received the requesting Business Document.
- Verify content integrity – Verify the integrity of the original content of the Business Document request.

**isPositiveResponse**

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

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

As mentioned earlier, this mechanism is very important in expressing OAGI scenarios as ebXML Collaboration Definitions because it allows one to specify the intent of the BOD (success or failure) and consequently provide Collaboration Definitions that can be reused beyond OAGI document format definitions.

**ebXML Signals**

Business signals are application-level documents that "signal" the current state of the Business Transaction. These business signals have specific business purposes and are separate from lower protocol and transport signals.

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

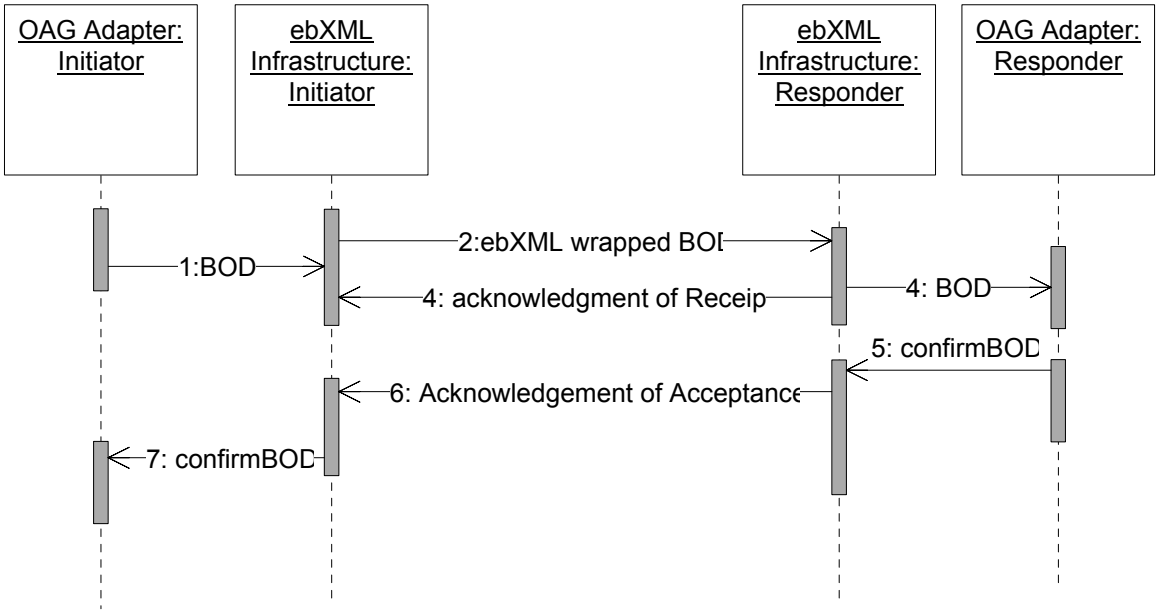
The Business Process Specification Schema provides both the choreography of business signals and the structure definition of the business payload of a business signal. The ebXML Message Service Specification signal structures provide business service state alignment infrastructure, including unique message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

A DTD is provided for each of the possible business signals in section 9 of the ebXML BPSS document.

This design was borrowed from the RosettaNet Implementation Framework specification. Consequently, the section of the OAGI RosettaNet IF 2.0 white paper was used almost as is for the purpose of this section.

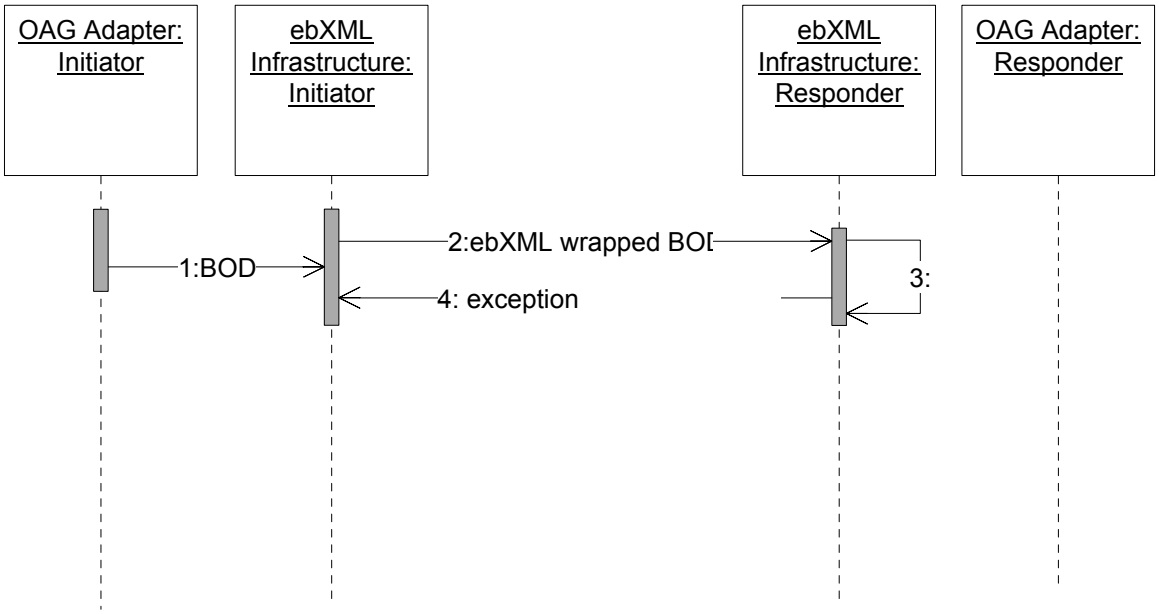
The following diagrams show message exchange sequences for the use of ebXML Signal Messages for specific error conditions. In a "discrete system model" when a ConfirmBOD is used, the STATUSLVL field value determines which ebXML Signal Message is used. The use of the ConfirmBOD is controlled through the CONFIRMATION field in the CNTROLAREA and should be coordinated with the PIP requirements for the use of AcknowledgementOfReceipt.

Sequence Diagram 1: Communication with no error -



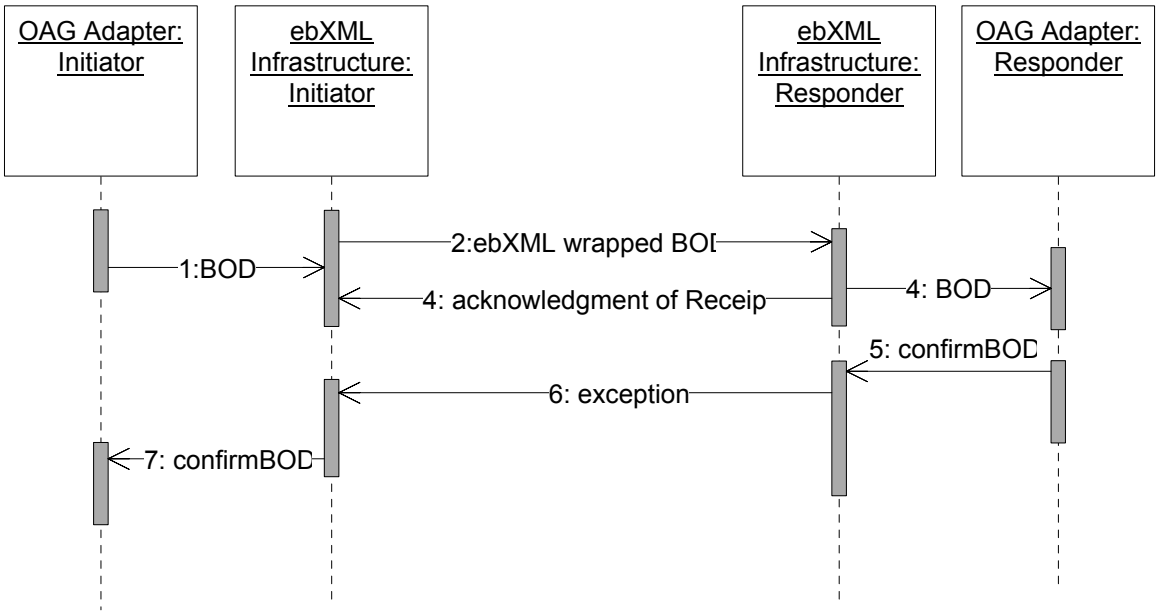
Receiving ebXMLAdapter (Responder) passes parser-level validation, OAGAdapter validates content and generates a ConfirmBOD with STATUSLVL="00" resulting in a non-substantive AcknowledgementOfAcceptance message. Sending ebXMLAdapter (Initiator) generates ConfirmBOD from Signal Message.

Sequence Diagram 2: Syntax Error – Exception: *Receipt-Acknowledgement-Exception*




Receiving ebXMLAdapter (Responder) detects a parser-level error and generates Exception: *Receipt-Acknowledgement-Exception*. Sending ebXMLAdapter (Initator) generates a ConfirmBOD with STATUSLVL="99" to indicate the error.

Sequence Diagram 3: Content Error – Exception: *Business-Exception*



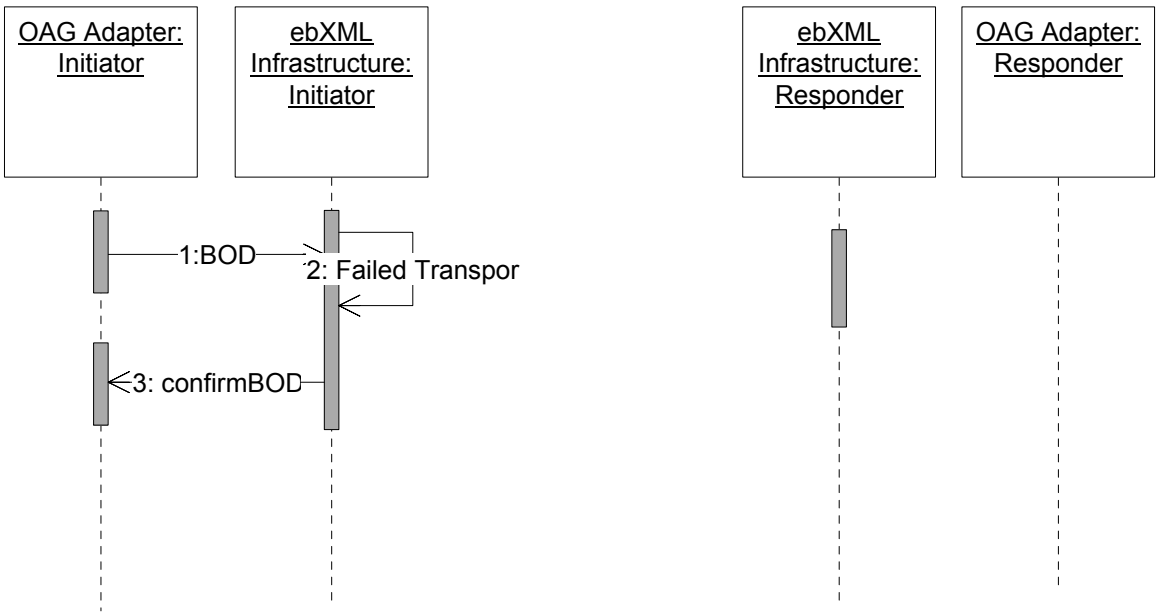
Receiving ebXMLAdapter (Responder) passes parser-level validation, OAGAdapter detects an OAGI content error and generates a ConfirmBOD with STATUSLVL="99". The receiver-side ebXMLAdapter (Responder) creates an Exception: *Business-Exception*. Sending ebXMLAdapter (Responder) re-creates a ConfirmBOD with STATUSLVL="99" to indicate the error.

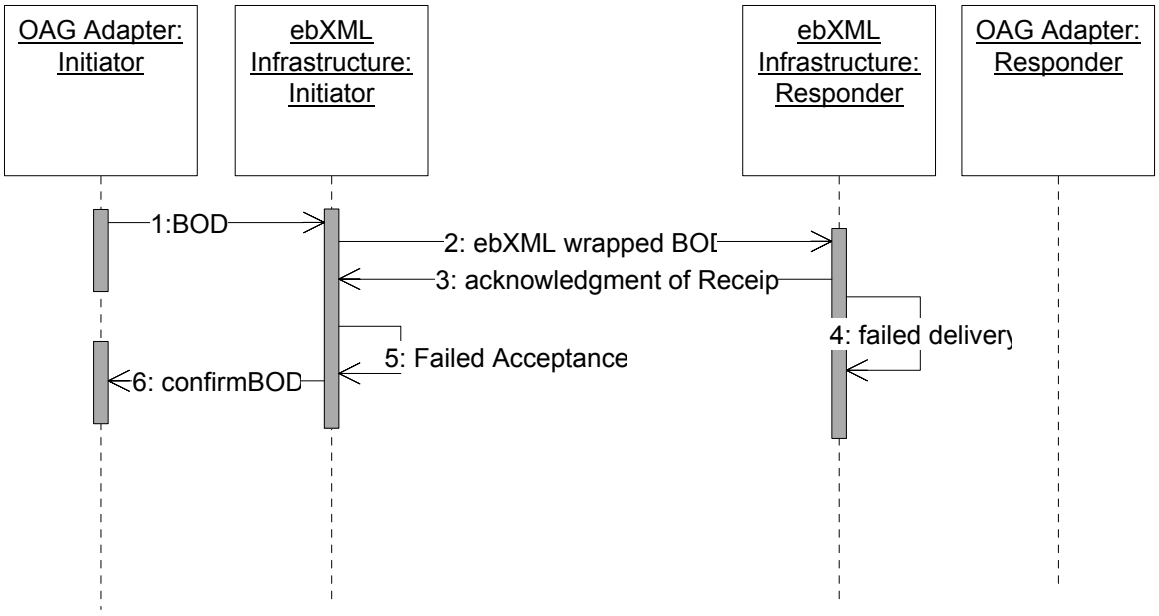
## RosettaNet Notification of Failure Message and ebXML

Unlike RosettaNet, ebXML does not provide a specific notification of failure to enable the initiator of a Collaboration to cancel a Business Transaction he has initiated without waiting for the response of the Responder.

Instead, ebXML provides a general mechanism to initiate a new Business Transaction while one is currently on-going. (See the onInitiation attribute of a Business Activity). If this mechanism is more generic and can be used for purposes other than aborting a transaction, it also has its drawback since it is required to be explicit in the Collaboration Definition.

Sequence Diagram 4: Communication-Level Failures






The illustrations in Diagram 4 show the message sequence when the ebXML infrastructure fails to transport the ebXML-wrapped message or fails to deliver the message to the OAGI-based recipient. This may be a result of transport-level errors or timeout conditions as defined in the Business Transaction. In the case of a timeout condition as a result of Failed Delivery (3), only the acknowledgment of receipt is returned by ebXMLAdapter. Unlike RosettaNet, the initiating ebXMLAdapter does not have to resend the message in this case; it is only the acceptance that failed, not the transport.

**Patterns**

ebXML Business Service Interfaces are configured to execute the business processes specified in a *Business Process Specification*. They do so by exchanging ebXML messages and business signals.

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

The Business Transaction Interaction Patterns set forth in Chapter 8 of the UMM N090R9.1 document illustrate recommended permutations of message sequences as determined by the type of Business Transaction defined and the timing policies specified in the transactions.

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

**Multiparty Collaborations**

Scenario diagrams are often composed of several Binary Collaborations that are interleaved with one another just like Scenario 55, which is composed of two Binary Collaborations that will occur in parallel and are dependent on one another.

The ebXML specification provides a way not only to express the two Binary Collaborations independently but also to synthesize them into a global “multiparty Collaboration.” Sequencing rules may also be defined to choreograph the Business Transactions across Binary Collaborations, such as “if Business Transaction 3 ends in Collaboration A, Business Transaction 2 can start in Collaboration B.”

The example below illustrates the fact that the intermediary is only allowed to send a cancel RFQ to the supplier when the buyer has itself successfully completed a Cancel RFQ transaction.

```
<MultiPartyCollaboration name="OAGI:RFQ / Quote With Intermediary">
        <BusinessPartnerRole name="Buyer">
                <Performs initiatingRole="Buyer"/>
                <Transition fromBusinessState="Cancel RFQ Buyer"
                            toBusinessState="Cancel RFQ Intermediary"/>
        </BusinessPartnerRole>
        …
```

## Application-to-Application Collaboration Definitions

People implementing the OAGI specification may also be interested in using a formal definition for the scenario diagrams that occur only in the Application-to-Application (A2A) scenario. The goal of this section is to define a precise subset of the ebXML specification that does not feature "business related" semantics to specify a long-running message interchange between two applications of the same company.

The benefit of this approach is that it provides a formal way to configure both application and interapplication infrastructure. This should prove useful when one needs to extend or modify existing implementations. Today, since there is no formal expression of the scenario diagrams, one has to basically hard code the scenario in some application or come up with a proprietary machine-readable representation of the scenario. In this section, we explore how to develop a "standard" representation based on the ebXML BPSS specification.

### The ebXML Metamodel Subset

The subset was deduced very simply by removing all business-related semantics (Table 2 presents the major changes to the metamodel).

As one may have expected, most of the changes have been at the Business Transaction level. The sequencing rules remain unchanged as well as the multiparty Collaboration design. Partner Roles have been replaced by component roles, such as Inventory or Billing.

Signals are much simpler than in ebXML. Receipts are somewhat useless in the Application-to-Application scenario. OAGI already provides a signal that can be used for both acceptance acknowledgement and exceptions (ConfirmBOD). We have left the *timeToAcknowledgeAcceptance* attribute to a requesting activity in order to indicate that a CONFIRMBOD signal is expected.

The following figure presents a simple subset of the ebXML BPSS Metamodel designed for Application to Application integration only.

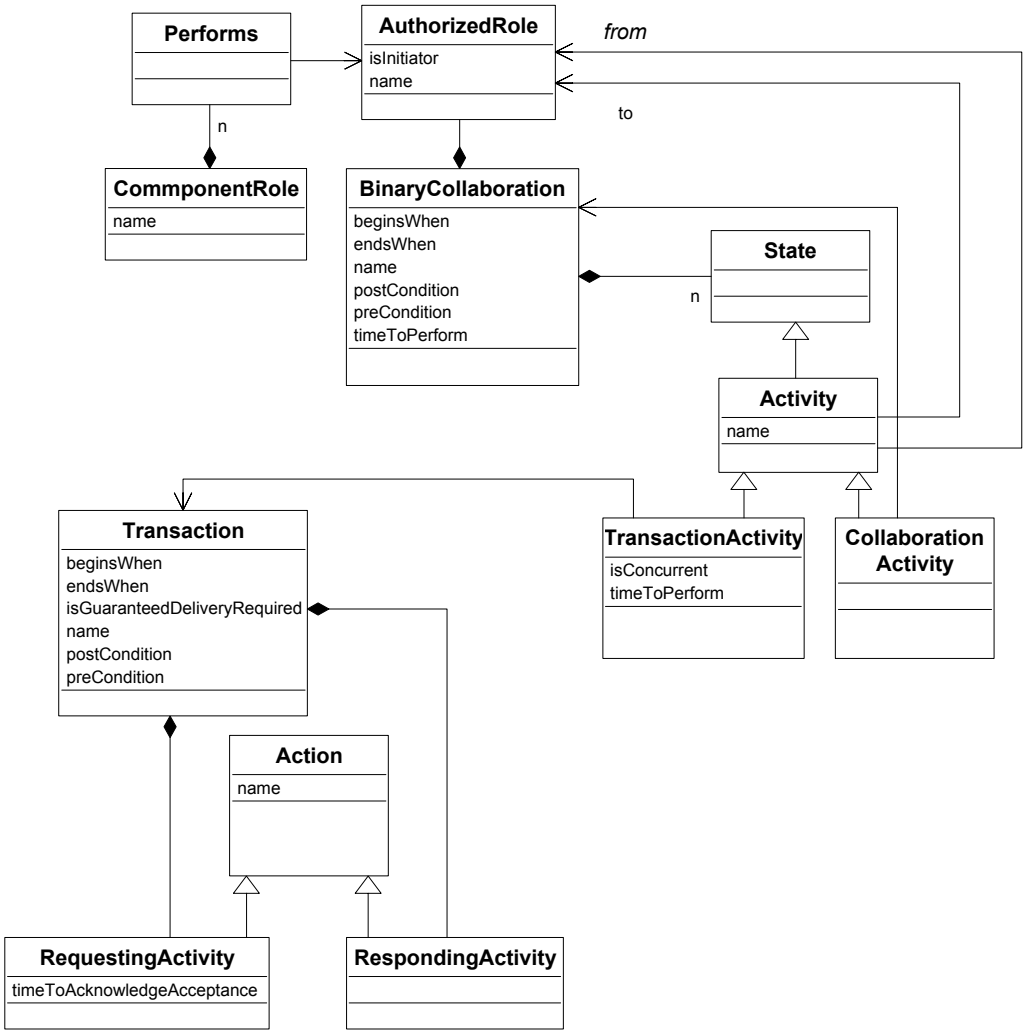



Figure 114. ebXML Metamodel Subset for A2A Integration Scenarios

Table 2. ebXML Semantics[1] Applied to Application-to-Application Scenarios

| Parameter | A2A | Comments |
|---|---|---|
| ***Business Transaction*** | ***Transaction*** | |
| Pattern | No | |
| isGuaranteedDeliveryRequired | Possible | |
| precondition | Yes | Optional |
| PostCondition | Yes | Optional |
| BeginsWhen | Yes | Optional |
| EndWhen | Yes | Optional |
| | | |
| ***Business Transaction Activity*** | ***TransactionActivity*** | |
| TimeToPerform | Yes | Optional |
| isLegallyBinding | No | |
| IsConcurrent | Yes | Optional |
| | | |
| ***Business Action*** | ***Action*** | |
| IsIntelligibleCheckRequired | No | |
| IsAuthorizationRequired | No | |
| timeToAcknowledgeReceipt | No | |
| isNonRepudiationRequired | No | |
| isNonRepudiationOfRecieptRequired | No | |
| timeToAcknowledgeAcceptance | Yes | Optional |
| | | |
| ***DocumentEnvelope*** | Idem | |
| isPositiveResponse | Yes | Mandatory |
| | | |
| ***DocumentSecurity*** | Idem | |
| IsTamperProof | Yes | Optional |
| IsConfidential | Yes | Optional |
| | | |
| ***Binary Collaboration*** | Idem | |
| precondition | Yes | Optional |
| PostCondition | Yes | Optional |
| BeginsWhen | Yes | Optional |
| EndWhen | Yes | Optional |
| TimeToPerform | Yes | Optional |
| Pattern | No | |
| | | |

### DTD

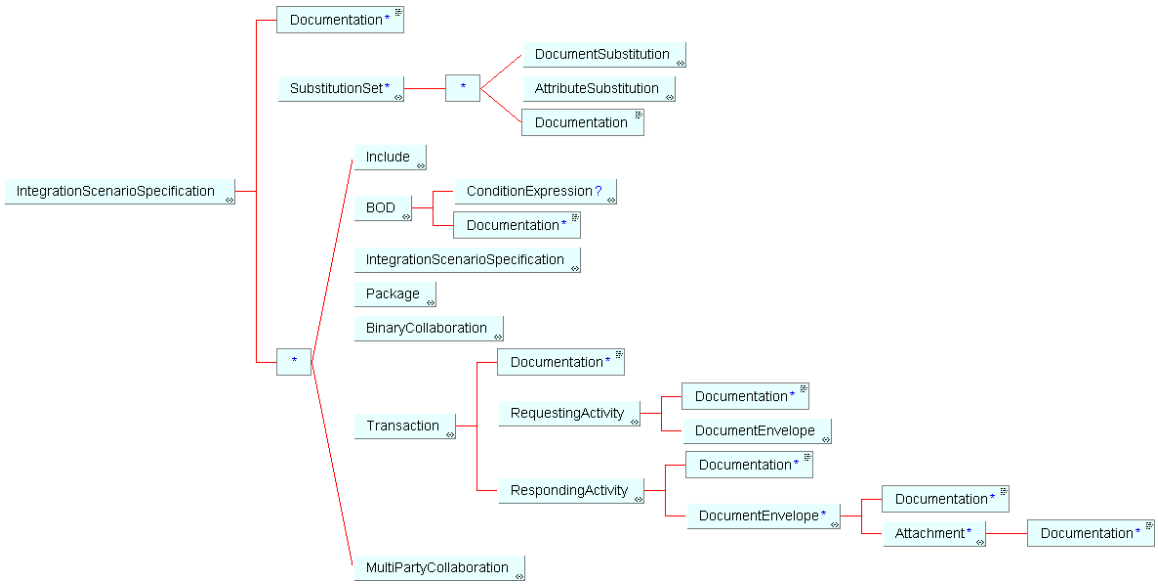
The corresponding DTD was created from the one of ebXML.

```
<!-- ============================================================ -->
<!--  Editor:   Jean-Jacques Dubray (eXcelon Corp)                -->
<!--  Version:  Version 1.00                                       -->
<!--  Updated:  2001-06-15                                        -->
<!--                                                               -->
<!--  Public Identifier:                                           -->
<!--            "-//OAGI//DTD Integration Scenarion Specification ver
1.0//EN"    -->
<!--                                                               -->
<!--                                                               -->
<!-- ============================================================ -->
<!ELEMENT IntegrationScenarioSpecification  (Documentation* ,
SubstitutionSet* ,  (Include | BOD | IntegrationScenarioSpecification |
Package | BinaryCollaboration | Transaction | MultiPartyCollaboration )* )>
<!ATTLIST IntegrationScenarioSpecification  name    ID     #REQUIRED
                                           uuid    CDATA  #REQUIRED
                                           version CDATA  #REQUIRED >
<!ELEMENT Documentation  (#PCDATA )>
<!ATTLIST Documentation  uri CDATA  #IMPLIED >
<!ELEMENT Include  (Documentation* )>
<!ATTLIST Include  name    CDATA  #REQUIRED
                   uuid    CDATA  #REQUIRED
                   uri     CDATA  #REQUIRED
                   version CDATA  #REQUIRED >
<!ELEMENT BOD  (ConditionExpression? , Documentation* )>
<!ATTLIST BOD  name                 CDATA  #REQUIRED
               nameID               ID     #IMPLIED
               specificationLocation CDATA  #IMPLIED
               specificationElement  CDATA  #IMPLIED >
<!ELEMENT ConditionExpression  (Documentation* )>
<!ATTLIST ConditionExpression  expressionLanguage CDATA  #IMPLIED
                               expression         CDATA  #IMPLIED >
<!ELEMENT SubstitutionSet  (DocumentSubstitution | AttributeSubstitution |
Documentation )*>
<!ATTLIST SubstitutionSet  name         CDATA  #IMPLIED
                           nameId       IDREF  #IMPLIED
                           applyToScope CDATA  #IMPLIED >
<!ELEMENT DocumentSubstitution  (Documentation* )>
<!ATTLIST DocumentSubstitution  originalBOD     CDATA  #IMPLIED
                                originalBODID   CDATA  #IMPLIED
                                substituteBOD   CDATA  #IMPLIED
                                substituteBODId CDATA  #IMPLIED >
<!ELEMENT AttributeSubstitution  (Documentation* )>
<!ATTLIST AttributeSubstitution  attributeName CDATA  #IMPLIED
                                 value         CDATA  #IMPLIED >
<!ELEMENT Package  (Documentation* ,  (Package | BinaryCollaboration |
Transaction | MultiPartyCollaboration )* )>
<!ATTLIST Package  name   CDATA  #REQUIRED
                  nameID ID     #IMPLIED >
<!ELEMENT BinaryCollaboration  (Documentation* , InitiatingRole ,
RespondingRole ,  (Documentation | Start | Transition | Success | Failure |
TransactionActivity | CollaborationActivity | Fork | Join )* )>
<!ATTLIST BinaryCollaboration  name          CDATA  #REQUIRED
```

```
                              nameID        ID     #IMPLIED
                              beginsWhen    CDATA  #IMPLIED
                              endsWhen      CDATA  #IMPLIED
                              preCondition  CDATA  #IMPLIED
                              postCondition CDATA  #IMPLIED
                              timeToPerform CDATA  #IMPLIED >
<!ELEMENT MultiPartyCollaboration  (Documentation* , ComponentRole* )>
<!ATTLIST MultiPartyCollaboration  name   CDATA  #REQUIRED
                                   nameID ID     #IMPLIED >
<!ELEMENT InitiatingRole  (Documentation* )>
<!ATTLIST InitiatingRole  name   CDATA  #REQUIRED
                          nameID ID     #IMPLIED >
<!ELEMENT RespondingRole  (Documentation* )>
<!ATTLIST RespondingRole  name   CDATA  #REQUIRED
                          nameID ID     #IMPLIED >
<!-- A BusinessState is one of Start, Success, Failure, Fork, Join,
BusinessTransactionActivity or CollaborationActivity -->
<!-- fromBusinessState and toBusinessState are fully qualified using XPath --
>
<!ELEMENT Transition  (ConditionExpression? , Documentation* )>
<!ATTLIST Transition  onInitiation            (true | false )  'false'
                     fromBusinessState      CDATA  #IMPLIED
                     fromBusinessStateIDRef IDREF  #IMPLIED
                     toBusinessState        CDATA  #IMPLIED
                     toBusinessStateIDRef   IDREF  #IMPLIED
                     conditionGuard          (Success |
                                              BusinessFailure |
                                              TechnicalFailure |
                                              AnyFailure )  #IMPLIED >
<!-- Start is a special type of Transition in that it only has a destination
-->
<!ELEMENT Start  (Documentation* )>
<!ATTLIST Start  toBusinessState      CDATA  #REQUIRED
                 toBusinessStateIDRef IDREF  #IMPLIED >
<!-- Success is a special type of Transition in that it only has a
origination -->
<!ELEMENT Success  (ConditionExpression? , Documentation* )>
<!ATTLIST Success  fromBusinessState      CDATA  #REQUIRED
                   fromBusinessStateIDRef IDREF  #IMPLIED
                   conditionGuard          (Success |
                                            BusinessFailure |
                                           TechnicalFailure |
                                            AnyFailure )  #IMPLIED >
<!-- Failure is a special type of Transition in that it only has a
origination -->
<!ELEMENT Failure  (ConditionExpression? , Documentation* )>
<!ATTLIST Failure  fromBusinessState      CDATA  #REQUIRED
                   fromBusinessStateIDRef IDREF  #IMPLIED
                   conditionGuard          (Success |
                                            BusinessFailure |
                                            TechnicalFailure |
                                            AnyFailure )  #IMPLIED >
<!-- Fork is a special type of BusinessState that can be transitioned to -->
<!ELEMENT Fork  (Documentation* )>
<!ATTLIST Fork  name   CDATA  #REQUIRED
                nameID ID     #IMPLIED >
<!-- Join is a special type of BusinessState that can be transitioned to -->
<!ELEMENT Join  (Documentation* )>
```

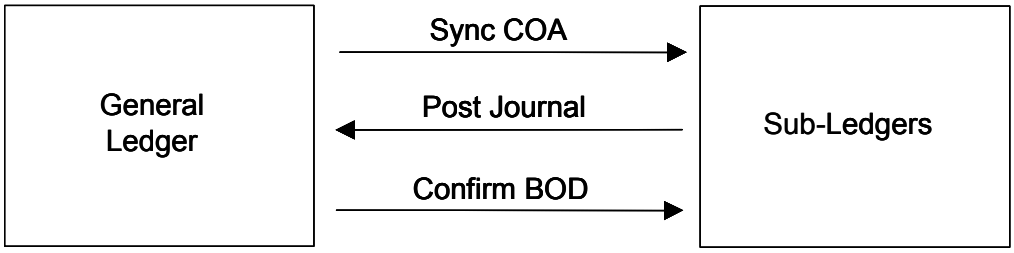
```
<!ATTLIST Join  name       CDATA  #REQUIRED
               nameID     ID     #IMPLIED
                waitForAll  (true | false )  'true' >
<!-- fromAuthorizedRole and toAuthorizedRole are fully qualified using XPath
-->
<!-- BusinessTransactionActivity is a BusinessState that can be transitioned
to -->
<!ELEMENT TransactionActivity  (Documentation* )>
<!ATTLIST TransactionActivity  name                    CDATA  #REQUIRED
                               nameID                  ID     #IMPLIED
                               fromAuthorizedRole      CDATA  #REQUIRED
                               fromAuthorizedRoleIDRef IDREF  #IMPLIED
                               toAuthorizedRole        CDATA  #REQUIRED
                               toAuthorizedRoleIDRef   IDREF  #IMPLIED
                               isConcurrent             (true | false )
'true'
                               timeToPerform           CDATA  #IMPLIED
                               transaction             CDATA  #IMPLIED
                               transactionIDRef        CDATA  #IMPLIED >

<!-- fromAuthorizedRole and toAuthorizedRole are fully qualified using XPath
-->
<!-- CollaborationActivity is a BusinessState that can be transitioned to -->
<!ELEMENT CollaborationActivity  (Documentation* )>
<!ATTLIST CollaborationActivity  name                    CDATA  #REQUIRED
                                 nameID                  ID     #IMPLIED
                                 fromAuthorizedRole      CDATA  #REQUIRED
                                 fromAuthorizedRoleIDRef IDREF  #IMPLIED
                                 toAuthorizedRole        CDATA  #REQUIRED
                                 toAuthorizedRoleIDRef   IDREF  #IMPLIED
                                 binaryCollaboration     CDATA  #REQUIRED
                                 binaryCollaborationIDRef IDREF  #IMPLIED >
<!ELEMENT Transaction  (Documentation* , RequestingActivity ,
RespondingActivity )>
<!ATTLIST Transaction  name                         CDATA  #REQUIRED
                       nameID                       ID     #IMPLIED
                       beginsWhen                   CDATA  #IMPLIED
                       endsWhen                     CDATA  #IMPLIED
                       isGuaranteedDeliveryRequired  (true | false )  'false'
                       preCondition                 CDATA  #IMPLIED
                       postCondition                CDATA  #IMPLIED >
<!ELEMENT RequestingActivity  (Documentation* , DocumentEnvelope )>
<!ATTLIST RequestingActivity  name                         CDATA  #IMPLIED
                              nameID                       ID     #IMPLIED
                               timeToAcknowledgeAcceptance CDATA  #IMPLIED >
<!ELEMENT RespondingActivity  (Documentation* , DocumentEnvelope* )>
```

```
<!ATTLIST RespondingActivity  name   CDATA  #IMPLIED
                              nameID ID     #IMPLIED >
<!ELEMENT DocumentEnvelope  (Documentation* , Attachment* )>
<!ATTLIST DocumentEnvelope  isPositiveResponse  (true | false )  'false'
                            isAuthenticated     (true | false )  'false'
                            isConfidential      (true | false )  'false'
                            isTamperProof       (true | false )  'false'
                            BOD                 CDATA  #IMPLIED
                            BODIDRef            CDATA  #IMPLIED >
<!ELEMENT Attachment  (Documentation* )>
<!ATTLIST Attachment  name            CDATA  #REQUIRED
                      nameID          ID     #IMPLIED
                      mimeType        CDATA  #REQUIRED
                      specification   CDATA  #IMPLIED
                      version         CDATA  #IMPLIED
                      isAuthenticated  (true | false )  'false'
                      isConfidential   (true | false )  'false'
                      isTamperProof    (true | false )  'false'
                      BOD             CDATA  #IMPLIED
                      BODIDRef        CDATA  #IMPLIED >
<!ELEMENT ComponentRole  (Documentation* , Performs* , Transition* )>
<!ATTLIST ComponentRole  name   CDATA  #REQUIRED
                         nameID ID     #IMPLIED >
<!-- authorizedRole is fully qualified using XPath -->
<!ELEMENT Performs  (Documentation* )>
<!ATTLIST Performs  initiatingRole        CDATA  #IMPLIED
                    inititiatingRoleIDRef IDREF  #IMPLIED
                    respondingRole        CDATA  #IMPLIED
                    respondingRoleIDRef   IDREF  #IMPLIED >
```


IntegrationScenarioSpecification
Documentation*
SubstitutionSet*
*
DocumentSubstitution
AttributeSubstitution
Documentation
*
Include
BOD
ConditionExpression?
Documentation*
IntegrationScenarioSpecification
Package
BinaryCollaboration
Transaction
Documentation*
RequestingActivity
Documentation*
DocumentEnvelope
RespondingActivity
Documentation*
DocumentEnvelope*
Documentation*
Attachment*
Documentation*
MultiPartyCollaboration

**Example**

Integration Scenario 1.0 describes the integration scenario for subledger business software components to integrate with a general ledger business software component because many applications create data that causes changes in the account balances of a general ledger application.




Accounts Payable
Accounts Receivable
Budget
Project Accounting
Manufacturing
Inventory
Order Management
Billing
Purchasing
Assets
Human Resources

We can organize this simple scenario into two transactions (Sync COA and Post Journal/Confirm BOD as an acceptance signal).

The XML integration scenario definition follows.

```
<!--
 edited by Jean-Jacques Dubray (eXcelon Corp.)
 -->
<!--
 Notes
 -->
<IntegrationScenario name="1.0 GENERAL LEDGER TO SUB-LEDGER SCENARIO" version="1.0"
      uuid="[1234-5678-901234]">
   <!--
    Business Documents
    -->
    <BusinessDocument name="Sync COA" />
    <BusinessDocument name="Post Journal" />
   <Package name="OAGIS">
      <!--
       Binary Collaboration
       -->
       <BinaryCollaboration name="BC:GENERAL LEDGER TO SUB-LEDGER SCENARIO"
            timeToPerform="P1H">
          <Documentation>timeToPerform = Period: 1 hour from start of
               transaction</Documentation>
          <InitiatingRole name="Ledger" />
          <RespondingRole name="Sub-ledger" />
          <BusinessTransactionActivity name="Sync COA" businessTransaction="BT:Sync
               COA" fromAuthorizedRole="Ledger" toAuthorizedRole="Sub-Ledger" />
          <BusinessTransactionActivity name="Post Journal" businessTransaction="BT:Post
               Journal" fromAuthorizedRole="Sub-Ledger" toAuthorizedRole="Ledger" />
          <Start toBusinessState="Sync COA" />
          <Transition fromBusinessState="Sync COA" toBusinessState="Post Journal"
               conditionGuard="Success" />
          <Success fromBusinessState="Post Journal" guardCondition="success" />
          <Failure fromBusinessState="Post Journal" guardCondition="BusinessFailure"
               guardExpression="//CONFIRMBOD/CODE="Reject"" />
       </BinaryCollaboration>
       <!--
       Here are all the Business Transactions needed
       -->
       <BusinessTransaction name="BT:Sync COA">
          <RequestingBusinessActivity name="">
             <DocumentEnvelope businessDocument="Sync COA" />
          </RequestingBusinessActivity>
       </BusinessTransaction>
       <BusinessTransaction name="BT:Post Journal">
          <RequestingBusinessActivity name="" timeToAcknowledgeAcceptance="P10M">
             <DocumentEnvelope isPositiveResponse="true" businessDocument="" />
          </RequestingBusinessActivity>
       </BusinessTransaction>
   </Package>
 </IntegrationScenario>
```

# ebXML Collaboration Protocol Profile and Agreement

As defined in the ebXML Business Process Specification Schema [ebBPSS], a Business Partner is an entity that engages in Business Transactions with another Business Partner(s) [1]. Each Partner's capabilities (both commercial/Business and technical) to engage in electronic Message exchanges with other Partners MAY be described by a document called a Trading-Partner Profile (TPP). The agreed interactions between two Partners MAY be documented in a document called a Trading-Partner Agreement (TPA). A TPA MAY be created by computing the intersection of the two Partners' TPPs.

The Message-exchange capabilities of a Party MAY be described by a Collaboration Protocol Profile (CPP) within the TPP. The Message-exchange agreement between two Parties MAY be described by a Collaboration Protocol Agreement (CPA) within the TPA. Included in the CPP and CPA are details of transport, messaging, security constraints, and bindings to a Business Process Specification (or, for short, Process Specification) document that contains the definition of the interactions between the two Parties while engaging in a specified electronic Business Collaboration.

The goal of this section is to provide guidelines in developing CPP s and CPAs that support OAGIS Business Collaborations.

## Collaboration Protocol Profile

The CPP contains the information relative to the Party (***PartyInfo*** element) that owns the CPP as well as how the Message Header and payload constituent(s) are packaged for transmittal (Packaging element).

The OAGI guidelines are featured in Table 3. Most of the constraints are at the delivery channel and reliable messaging levels.

Table 3. CPP Guidelines

| EbXML PartyInfo Elements | Action for OAGI |
|---|---|
| **PartyId** @type | Follow ebXML guidelines |
| **PartyRef** @href | Follow ebXML guidelines |
| **CollaborationRole** | N/A |
| **ProcessSpecification** @version @name @xlink:type @xlink:href | Pointer to an OAGI Collaboration Definition |
| **Role** @name @xlink:type @xlink:href | Role played by the Party within the Collaboration |
| **CertificateRef** @certId | Follow ebXML guidelines |
| **ServiceBinding** @channelId @packageId | Default service binding - Follow ebXML guidelines |
| **Service** @type | |
| **Override** @action @channelId @packageId @xlink:href @xlink:type | If any override binding is needed (e.g. security can be added). Follow ebXML |

| | |
|---|---|
| | guidelines |
| **Certificate** @certId | Follow ebXML guidelines |
| **KeyInfo** | Follow ebXML guidelines |
| **DeliveryChannel** @channeled @transported @docExchangeId | Follow ebXML guidelines |
| **Characteristics** @syncReplyMode | Following OAGI model the Business-response Message should be asynchronous; however, the Party may still use a synchronous signal |
| @nonrepudiationOfOrigin | Based on the OAGI scenario, this parameter may be set either way |
| @nonrepudiationOfReceipt | Based on the OAGI scenario, this parameter may be set either way |
| @secureTransport | Based on the OAGI scenario, this parameter may be set either way |
| @confidentiality | Based on the OAGI scenario, this parameter may be set either way |
| @authenticated | Based on the OAGI scenario, this parameter may be set either way |
| @authorized | Based on the OAGI scenario, this parameter may be set either way |
| **Transport** @transported<br>**SendingProtocol** @version<br>**ReceivingProtocol** @version<br>**Endpoint** @uri @type<br>**TransportSecurity**<br>**Protocol** @version<br>**CertificateRef** @certId | Follow ebXML guidelines |
| **DocExchange** @docExchangeId | N/A |
| **ebXMLBinding** @version | N/A |
| **ReliableMessaging**<br>@deliverySemantics<br>@idempotency<br>@messageOrderSemantics<br>**Retries**<br>**RetryInterval**<br>**PersistDuration** | Messages should be delivered "*OnceAndOnlyOnce",* with an idempotency test (check for duplicates) and a guaranteed order. Retries, and retry interval and persist duration can be set to any appropriate value |
| **NonRepudiation**<br>**Protocol**<br>**HashFunction**<br>**SignatureAlgorithm**<br>**CertificateRef** @certId | Follow ebXML guidelines |
| **DigitalEnvelope**<br>**Protocol** @version<br>**EncryptionAlgorithm**<br>**CertificateRef** @certId | Follow ebXML guidelines |

Typically the subtree under the Packaging element indicates the specific way in which constituent parts of the Message are organized. MIME processing capabilities are typically the capabilities or agreements described in this subtree. The Packaging element provides information about MIME content types, XML namespaces, security parameters, and MIME structure of the data that is exchanged between Parties.

The ***ProcessingCapabilities*** element specifies whether the Messaging Service may parse or generate the "packaging constructs." Presently, the BODs are generated by the application rather than the messaging service layer, hence the values provided in the table below.

***CompositeList*** is a container for the specific way in which the simple parts are combined into groups (MIME multiparts) or encapsulated within security-related MIME content-types. The ***CompositeList*** element MAY be omitted from Packaging when no security encapsulations or composite multiparts are used.

```
<CompositeList>
      <Composite mimetype = "multipart/related" id = "P13"
                 mimeparameters = "type=application/xml">
            <Constituent idref = "BOD1"/>
            <Constituent idref = "BOD2"/>
      </Composite>
</CompositeList>
```

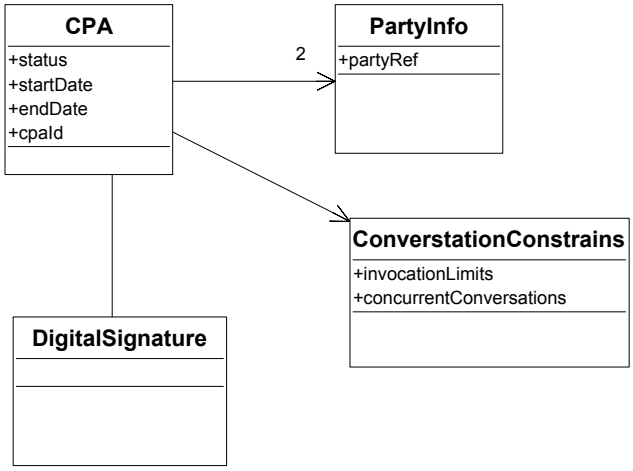
Table 4. ebXML Packaging Element

| EbXML Packaging Elements | Action for OAGI |
|---|---|
| **ProcessingCapabilities** @parse | true |
| @generate | false |
| **SimplePart** @mimetype | "application/xml" |
| **NamespaceSupported** @location<br>@version | Provide OAGI namespace reference |
| **CompositeList** | |
| **Composite** @mimetype | "multipart/related" |
| @mimeparameters | Value should be set to "type=application/xml" |
| **Constituent** @idref | Reference to the corresponding simplepart |

## Collaboration Protocol Agreement

A Collaboration Protocol Agreement (CPA) defines the capabilities that two Parties must agree upon to enable them to engage in electronic Business for the purposes of the particular CPA. In particular, the CPA is "calculated" as the intersection of two CPPs:

Figure 125. CPA Metamodel




The PartyInfo element has the same content as the corresponding CPP element, limited to the intersection of the two CPPs involved.

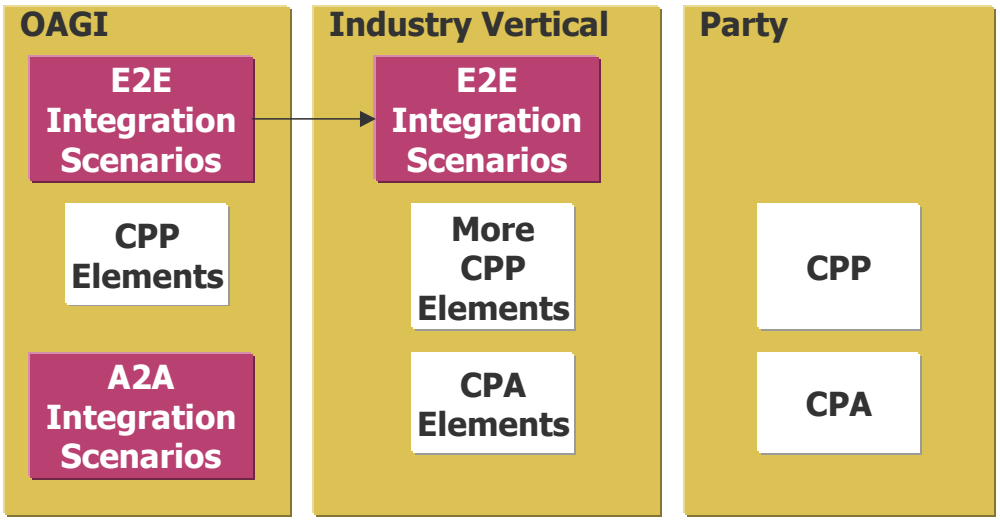
All the elements of the CPA can be used as is by the ebXML specification; there is no particular recommendation with respect to its usage in the context of OAGI.

# Bringing it All Together

## The Role of the OAGI Organization in Specifying CPPs and CPAs

Obviously, a lot of the parameters specified in the CPP and CPA are specific to companies and Parties involved in the Collaboration. The Role of OAGI is to specify both end-to-end and application-to-application scenarios, and in certain cases some CPP elements or constraints.

Figure 16. Role of Open Applications Group

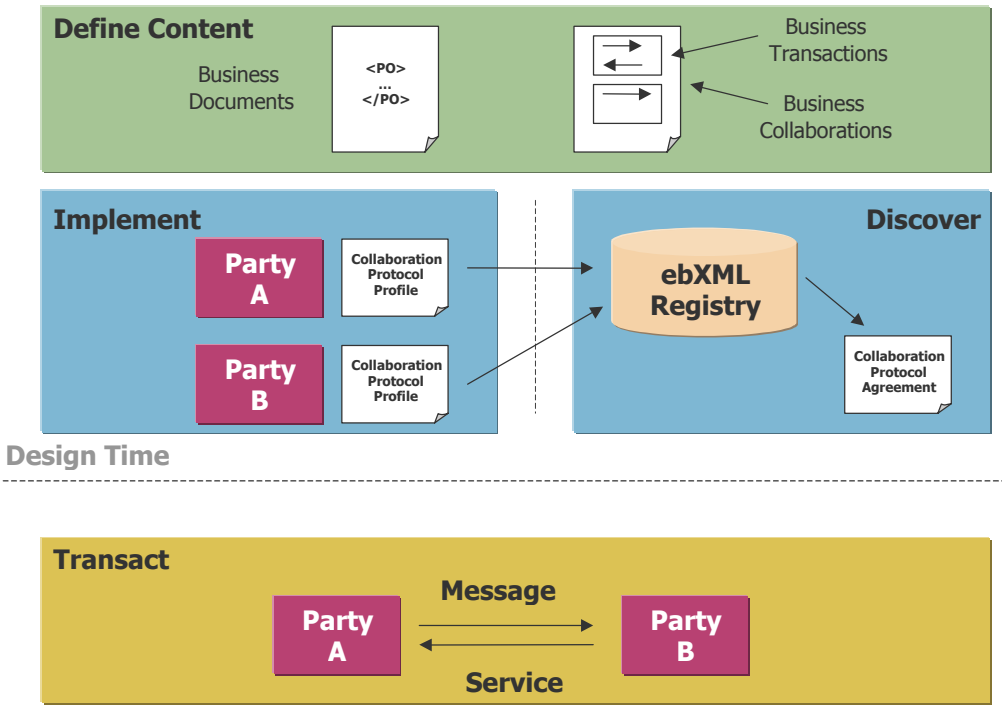



OAGI is also working actively with industry vertical consortia such as the STAR (Standard for Technology in the Automotive Retail) to specify specialize integration scenarios and BODs. These verticals provide more context to specify CPP and sometimes CPA elements. Finally, a user of an OAGI or Vertical OAGI specification will have to create complete CPPs and CPAs in order to carry out electronic commerce within the ebXML framework.

## ebXML Functional Phases

ebXML provides a complete framework to carry out electronic commerce in a near real-time, secure, and legally binding environment. Once Content, Collaborations, CPPs, and CPAs have been specified, the Parties involved can use the ebXML Messaging Service as is to carry out electronic transactions.

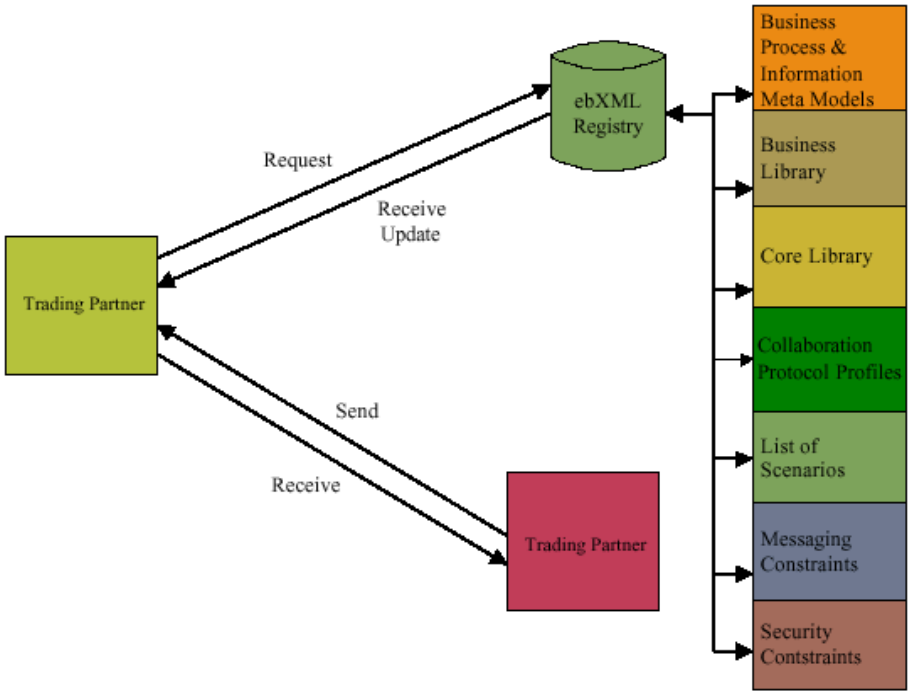
Figure 17. The four phases of a B2B implementation




In particular, and ebXML Registry is accessible by all and contains the following artifacts (see figure below):

- Business Process and information metamodels
- Business Library
- Core Library
- Collaboration Protocol Profiles
- List of scenarios
- Messaging constraints
- Security constraints

Figure 18. ebXML Registry Overview



# Appendix 1: ebXML Deliverables (www.ebXML.org)

There are four categories of ebXML deliverables:

- **Technical Specifications**
- **Technical Reports**
- **Reference Materials**
- **White Papers**

*Note:* the PDF versions of these documents are the normative versions.
Click **here** to download the PDF versions of all of the Specification documents. (4,880 kb zip file).
Click **here** to download the PDF versions of all of the Technical Report and Reference Material documents. (3,741 Kb zip file).
Click **here** to download the PDF versions of all of the White Paper documents. (338 Kb zip file).
These documents are also available from the XML.org Registry at **www.xml.org/registry**

### Technical Specifications

Technical Specifications are documents whose material fulfils the requirements of the **ebXML Requirements document.**
The following Technical Specification was approved by the ebXML Plenary on 16 February 2001.

| Specification | Project Team | Document |
|---|---|---|
| ebXML Technical Architecture Specification v1.04 | Technical Architecture | **ebTA.pdf** **ebTA.doc** |

The following Technical Specifications were approved by the ebXML Plenary on 11 May 2001.

| Specification | Project Team | Document |
|---|---|---|
| Business Process Specification Schema v1.01 (XML schema and DTD examples available separately) | Business Process | **ebBPSS.pdf** **ebBPSS.doc** **ebBPSS.dtd** **ebBPSS.xml** **ebBPSS.xsd** |
| Registry Information Model v1.0 | Registry/Repository | **ebRIM.pdf** **ebRIM.doc** |
| Registry Services Specification v1.0 | Registry/Repository | **ebRS.pdf** **ebRS.doc** |
| EbXML Requirements Specification v1.06 | Requirements | **ebREQ.pdf** **ebREQ.doc** |
| Collaboration Protocol Profile | Trading Partner | **ebCPP.pdf** |

| | | |
|---|---|---|
| and Agreement Specification v1.0 | | **ebCPP.doc** |
| Message Service Specification v1.0 | Transport, Routing, and Protocol | **ebMS.pdf**<br>**ebMS.doc** |

### Technical Reports

Technical Reports are documents that are either

- Guidelines: documents that contain information to guide in the interpretation or implementation of ebXML concepts.
- Catalogs: documents that contain foundation material based on ebXML Technical Specifications or Reports.

The following Technical Reports were accepted by the ebXML Plenary on 11 May 2001.

| Report | Project Team | Document |
|---|---|---|
| Business Process and Business Information Analysis Overview v1.0 | Business Process | bpOVER.pdf bpOVER.doc |
| Business Process Analysis Worksheets & Guidelines v1.0 | Business Process | bpWS.pdf bpWS.doc |
| E-Commerce Patterns v1.0 | Business Process | bpPATT.pdf bpPATT.doc |
| Catalog of Common Business Processes v1.0 | Business Process | bpPROC.pdf bpPROC.doc |
| Core Component Overview v1.05 | Core Components | ccOVER.pdf ccOVER.doc |
| Core Component Discovery and Analysis v1.04 | Core Components | ebCCD&A.pdf ebCCD&A.doc |
| Context and Re-Usability of Core Components v1.04 | Core Components | ebCNTXT.pdf ebCNTXT.doc |
| Guide to the Core Components Dictionary v1.04 | Core Components | ccCTLG.pdf ccCTLG.doc |
| Naming Convention for Core Components v1.04 | Core Components | ebCCNAM.pdf ebCCNAM.doc |
| Document Assembly and Context Rules v1.04 | Core Components | ebCCDOC.pdf ebCCDOC.doc |
| Catalogue of Context Drivers v1.04 | Core Components | ccDRIV.pdf ccDRIV.doc |
| Core Component Dictionary v1.04 | Core Components | ccDICT.pdf ccDICT.doc |
| Core Component Structure v1.04 | Core Components | ccSTRUCT.pdf ccSTRUCT.xls |
| Technical Architecture Risk Assessment v1.0 | Security | secRISK.pdf secRISK.doc |

### Reference Materials

Reference Materials are documents that are normative references in approved specifications.
The following Reference Material documents were accepted by the ebXML Plenary on 11 May 2001.

| Reference | Project Team | Document |
|---|---|---|
| ebXML Glossary | Technical Architecture | ebGLOSS.pdf<br>ebGLOSS.doc |

### White Papers

White Papers are documents that constitute a snapshot of ongoing work within each respective Project Team and represent a report that has been approved by the Project Team.

The following White Papers were accepted by the ebXML Steering Committee on 10-11 May 2001.

| White Paper | Project Team | Document |
|---|---|---|
| Proposed revisions to Technical Architecture Specification v1.0.4 | Business Process | bpTAREV.pdf<br>bpTAREV.doc |
| Using UDDI to find ebXML Registry/Repository | Registry/Repository | rrUDDI.pdf<br>rrUDDI.doc |
| ebXML Registry Security Proposal | Security | secREG.pdf<br>secREG.doc |

# Appendix 2: OAGI Scenario 55 RFQ / Quote

This section features the complete example used in this document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProcessSpecification SYSTEM "ebBPSS-v1.01.dtd">
<ProcessSpecification name="OAGI:55" uuid="1" version="1.0">
 <BusinessDocument name="Getlist RFQ"

specificationLocation="www.openapplications.org/OAGIS/v7.1/148 getlist rfq 00
3.dtd"
                   specificationElement="GETLIST RFQ 003" />
 <BusinessDocument name="List RFQ"

specificationLocation="www.openapplications.org/OAGIS/v7.1/149 getlist rfq 00
3.dtd"
                   specificationElement="LIST_RFQ_003" />
 <BusinessDocument name="Get RFQ"

specificationLocation="www.openapplications.org/OAGIS/v7.1/150_get_rfq_003.dt
d"
                   specificationElement="GET RFQ 003" />
 <BusinessDocument name="Show RFQ"

specificationLocation="www.openapplications.org/OAGIS/v7.1/151 show rfq 003.d
td"
                   specificationElement="SHOW RFQ 003" />
 <BusinessDocument name="Cancel RFQ"

specificationLocation="www.openapplications.org/OAGIS/v7.1/146 cancel rfq 003
.dtd"
                   specificationElement="CANCEL_RFQ_003" />
 <BusinessDocument name="Respond RFQ"

specificationLocation="www.openapplications.org/OAGIS/v7.1/147_respond_rfq_00
3.dtd"
                   specificationElement="RESPOND RFQ 003" />
 <BusinessDocument name="Sync Quote"

specificationLocation="www.openapplications.org/OAGIS/v7.1/152 sync quote 003
.dtd"
                   specificationElement="SYNC QUOTE 003" />
 <BusinessDocument name="Add Quote"

specificationLocation="www.openapplications.org/OAGIS/v7.1/153 add quote 003.
dtd"
                   specificationElement="ADD_QUOTE_003" />
 <BusinessDocument name="Change Quote"

specificationLocation="www.openapplications.org/OAGIS/v7.1/154_change_quote_0
03.dtd"
                   specificationElement="CHANGE QUOTE 003" />
 <BusinessDocument name="Cancel Quote"
```

```
specificationLocation="www.openapplications.org/OAGIS/v7.1/155 cancel quote 0
03.dtd"
                  specificationElement="CANCEL QUOTE 003" />
 <BusinessDocument name="Respond Quote"

specificationLocation="www.openapplications.org/OAGIS/v7.1/156 respond quote
003.dtd"
                  specificationElement="RESPOND_QUOTE_003" />
 <BusinessDocument name="Elect Quote"

specificationLocation="www.openapplications.org/OAGIS/v7.1/002_confirm_bod_00
3.dtd"
                  specificationElement="CONFIRM BOD 003">
    <ConditionExpression expressionLanguage='XPath'
expression='//STATUSLVL="18"' />
    </BusinessDocument>



<Package name="IntermediarySupplier" nameID="Scenario 1">
        <BusinessTransaction name="BT:Getlist RFQ">
                <RequestingBusinessActivity
                        isNonRepudiationRequired="false"
                        timeToAcknowledgeReceipt="PT2H"
                        timeToAcknowledgeAcceptance="PT24H">
                        <DocumentEnvelope businessDocument="Getlist RFQ"/>
                </RequestingBusinessActivity>

                <RespondingBusinessActivity>
                        <DocumentEnvelope
                                businessDocument="list RFQ"
                                isPositiveResponse="true"/>
                </RespondingBusinessActivity>
        </BusinessTransaction>

        <BusinessTransaction name="BT:Get RFQ">
                <RequestingBusinessActivity
                        isNonRepudiationRequired="false"
                        timeToAcknowledgeReceipt="PT2H"
                        timeToAcknowledgeAcceptance="PT24H">
                        <DocumentEnvelope businessDocument="Get RFQ"/>
                </RequestingBusinessActivity>

                <RespondingBusinessActivity>
                        <DocumentEnvelope
                                businessDocument="Show RFQ"
                                isPositiveResponse="true"/>
                </RespondingBusinessActivity>
        </BusinessTransaction>

        <BusinessTransaction name="BT:Cancel RFQ">
                <RequestingBusinessActivity
                        isNonRepudiationRequired="false"
                        timeToAcknowledgeReceipt="PT2H"
                        timeToAcknowledgeAcceptance="PT24H">
```

```
                <DocumentEnvelope businessDocument="Cancel RFQ"/>
        </RequestingBusinessActivity>
```

```
</BusinessTransaction>

<BusinessTransaction name="BT:Respond RFQ">
        <RequestingBusinessActivity
                isNonRepudiationRequired="false"
                timeToAcknowledgeReceipt="PT2H"
                timeToAcknowledgeAcceptance="PT24H">
                <DocumentEnvelope businessDocument="Respond RFQ"/>
        </RequestingBusinessActivity>
        <RespondingBusinessActivity>
                <DocumentEnvelope
                        businessDocument="Respond RFQ"
                        isPositiveResponse="true"/>
        </RespondingBusinessActivity>
</BusinessTransaction>

<BusinessTransaction name="BT:Add Quote">
        <RequestingBusinessActivity
                isNonRepudiationRequired="true"
                timeToAcknowledgeReceipt="PT2H"
                timeToAcknowledgeAcceptance="PT24H">
                <DocumentEnvelope businessDocument="Add Quote"/>
        </RequestingBusinessActivity>
</BusinessTransaction>

<BusinessTransaction name="BT:Cancel Quote">
        <RequestingBusinessActivity
                isNonRepudiationRequired="true"
                timeToAcknowledgeReceipt="PT2H"
                timeToAcknowledgeAcceptance="PT24H">
                <DocumentEnvelope businessDocument="Cancel Quote"/>
        </RequestingBusinessActivity>
</BusinessTransaction>

<BusinessTransaction name="BT:Change Quote">
        <RequestingBusinessActivity
                isNonRepudiationRequired="true"
                timeToAcknowledgeReceipt="PT2H"
                timeToAcknowledgeAcceptance="PT24H">
                <DocumentEnvelope businessDocument="Change Quote"/>
        </RequestingBusinessActivity>
</BusinessTransaction>

<BusinessTransaction name="BT:Respond Quote">
        <RequestingBusinessActivity
                isNonRepudiationRequired="false"
                timeToAcknowledgeReceipt="PT2H"
                timeToAcknowledgeAcceptance="PT24H">
                <DocumentEnvelope businessDocument="Respond Quote"/>
        </RequestingBusinessActivity>
        <RespondingBusinessActivity>
                <DocumentEnvelope
                        businessDocument="Respond Quote"
                        isPositiveResponse="true"/>
```

```
        </BusinessTransaction>
        <BusinessTransaction name="BT:Elect Quote">

                <RequestingBusinessActivity

                        isNonRepudiationRequired="true"
                        timeToAcknowledgeReceipt="PT2H"
                        timeToAcknowledgeAcceptance="PT24H">
                        <DocumentEnvelope businessDocument="Elect Quote"/>
                </RequestingBusinessActivity>
        </BusinessTransaction>



 <BinaryCollaboration name="RFQ-QUOTE" timeToPerform="PT30D">
        <InitiatingRole name="supplier"/>
        <RespondingRole name="intermediary"/>
        <BusinessTransactionActivity name="BTA:Getlist RFQ"
                businessTransaction="BT:Getlist RFQ"
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>
        <BusinessTransactionActivity name="BTA:Get RFQ"
                businessTransaction="BT:Get RFQ"
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>
                    <BusinessTransactionActivity name="BTA:Add Quote"
                businessTransaction="BT:Add Quote"
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>
        <BusinessTransactionActivity name="BTA:Elect Quote"
                businessTransaction="BT:Elect Quote"
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>
        <CollaborationActivity binaryCollaboration="Review RFQ"
            fromAuthorizedRole="supplier" toAuthorizedRole="intermediary"/>
        <CollaborationActivity binaryCollaboration="Review Quote"
            fromAuthorizedRole="intermediary" toAuthorizedRole="supplier"/>
        <Start toBusinessState="Getlist RFQ"/>

        <Transition fromBusinessState="BTA:Getlist RFQ"
toBusinessState="BTA:Get RFQ"
                conditionGuard="Success"/>
        <Failure fromBusinessState="BTA:Getlist RFQ"
                conditionGuard="AnyFailure"/>
        <Transition fromBusinessState="BTA:Get RFQ"
toBusinessState="CA:Review RFQ"
                conditionGuard="Success"/>
        <Failure fromBusinessState="CA:Review RFQ"
                conditionGuard="AnyFailure"/>
        <Transition fromBusinessState="CA:Review RFQ"
toBusinessState="BTA:Add Quote"
                conditionGuard="Success"/>
        <Failure fromBusinessState="CA:Review RFQ"
                conditionGuard="AnyFailure"/>
        <Transition fromBusinessState="BTA:Add Quote"
toBusinessState="CA:Review Quote"
                conditionGuard="Success"/>
```

```
                conditionGuard="AnyFailure"/>
        <Transition fromBusinessState="CA:Review Quote"
toBusinessState="BTA:Elect Quote"
                conditionGuard="Success"/>

        <Failure fromBusinessState="CA:Review Quote"

                conditionGuard="AnyFailure"/>
        <Success fromBusinessState="BTA:Elect Quote"
conditionGuard="Success"/>
 </BinaryCollaboration>

 <BinaryCollaboration name="Review RFQ" timeToPerform="PT5D">
        <InitiatingRole name="supplier"/>
        <RespondingRole name="intermediary"/>
        <BusinessTransactionActivity name="BTA:Cancel RFQ"
                businessTransaction="BT:Cancel RFQ"
                fromAuthorizedRole="intermediary"
toAuthorizedRole="supplier"/>
        <BusinessTransactionActivity name="BTA:Respond RFQ Intermediary"
                businessTransaction="BT:Respond RFQ"
                fromAuthorizedRole="intermediary"
toAuthorizedRole="supplier"/>
        <BusinessTransactionActivity name="BTA:Respond RFQ Supplier"
                businessTransaction="BT:Respond RFQ"
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>

        <Fork name="Fork:Review RFQ Start"/>
        <Join name="Join:Review RFQ End" waitForAll="No"/>
        <Start toBusinessState="Fork:Review RFQ Start"/>

        <Transition fromBusinessState="Fork:Review RFQ Start"
toBusinessState="BTA:Respond RFQ Supplier"/>
        <Transition fromBusinessState="Fork:Review RFQ Start"
toBusinessState="BTA:Respond RFQ Intermdiary"/>
        <Transition fromBusinessState="Fork:Review RFQ Start"
toBusinessState="BTA:Cancel"/>
        <Transition fromBusinessState="BTA:Respond RFQ Intermediary"
toBusinessState="BTA:Respond RFQ Intermediary"/>
        <Transition fromBusinessState="BTA:Respond RFQ Supplier"
toBusinessState="BTA:Respond RFQ Supplier"/>
        <Failure fromBusinessState="BTA:Cancel RFQ"
                conditionGuard="Success"/>
                  <Success fromBusinessState="Join:Review RFQ End"/>
 </BinaryCollaboration>

 <BinaryCollaboration name="Review Quote" timeToPerform="PT5D">
        <InitiatingRole name="intermediary"/>
        <RespondingRole name="supplier"/>
                <BusinessTransactionActivity name="BTA:Cancel Quote"
                businessTransaction="BT:Cancel Quote"
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>
        <BusinessTransactionActivity name="BTA:Respond Quote"
                businessTransaction="BT:Respond Quote"
```

```
                fromAuthorizedRole="intermediary"
        <BusinessTransactionActivity name="BTA:Change Quote"
                businessTransaction="BT:Change Quote"
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>
        <BusinessTransactionActivity name="BTA:Sync Quote"
                businessTransaction="BT:Sync Quote"
```

```
                fromAuthorizedRole="supplier"
toAuthorizedRole="intermediary"/>
```

```
        <Fork name="Fork:Review Quote Start"/>
        <Join name="Join:Review Quote End" waitForAll="No"/>
                <Start toBusinessState="Fork:Review Quote Start"/>

        <Transition fromBusinessState="Fork:Review Quote Start"
toBusinessState="BTA:Respond Quote"/>
        <Transition fromBusinessState="Fork:Review Quote Start"
toBusinessState="BTA:SyncQuote"/>
        <Transition fromBusinessState="Fork:Review Quote Start"
toBusinessState="BTA:Cancel Quote"/>
        <Transition fromBusinessState="Fork:Review Quote Start"
toBusinessState="BTA:Change Quote"/>
        <Transition fromBusinessState="BTA:Respond Quote"
toBusinessState="BTA:Respond Quote"/>
        <Transition fromBusinessState="BTA:Change Quote"
toBusinessState="BTA:Change Quote"/>
        <Transition fromBusinessState="BTA:Sync Quote"
toBusinessState="BTA:Sync Quote"/>
        <Failure fromBusinessState="BTA:Cancel Quote"
                conditionGuard="Success"/>
           <Success fromBusinessState="Join:Review RFQ End"/>
 </BinaryCollaboration>

</Package>
</ProcessSpecification>
```

[1] Source ebXML BPSS Document