CSPRD02

ISO/IEC 19845

Edition csprd02
Version 01 November 2017
ccyy-mm-dd

Information Technology — Universal Business Language Version 2.2 (UBL v2.2)

Technologie d l'information — Langage d'affaire universel version 2.2 (UBL v2.2)

Reference number
ISO/IEC 19845:ccyy(E)

Contents

Page

List of Figures

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

The first version of ISO/IEC 19845 was prepared by the OASIS Universal Business Language Technical Committee [as OASIS Universal Business Language Version 2.1] and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, Information technology, in parallel with its approval by the national bodies of ISO and IEC. The content of ISO/IEC 19845:2015 and OASIS UBL v2.1 is identical.

This version of ISO/IEC 19845 is identical to UBL v2.2, a minor revision of UBL v2.1. Of significance, UBL v2.2 is fully backward compatible with UBL v2.1, that is, all schema-valid XML instances of ISO/IEC 19845:2015 are also schema-valid XML instances of this version of ISO/IEC 19845.

Introduction

Since its approval as a W3C recommendation in 1998, XML has been adopted in a number of industries as a framework for the definition of the messages exchanged in electronic commerce. The widespread use of XML has led to the development of multiple industry-specific XML versions of such basic documents as purchase orders, shipping notices, and invoices.

While industry-specific data formats have the advantage of maximal optimization for their business context, the existence of different formats to accomplish the same purpose in different business domains is attended by a number of significant disadvantages as well.

— Developing and maintaining multiple versions of common business documents like purchase orders and invoices is a major duplication of effort.

— Creating and maintaining multiple adapters to enable trading relationships across domain boundaries is an even greater effort.

— The existence of multiple XML formats makes it much harder to integrate XML business messages with back-office systems.

— The need to support an arbitrary number of XML formats makes tools more expensive and trained workers harder to find.

The OASIS Universal Business Language (UBL) is intended to help solve these problems by defining a generic XML interchange format for business documents that can be restricted or extended to meet the requirements of particular industries. Specifically, UBL provides the following:

— A suite of structured business objects and their associated semantics expressed as reusable data components and common business documents.

— A library of XML schemas for reusable data components such as “Address”, “Item”, and “Payment”—the common data elements of everyday business documents.

— A set of XML schemas for common business documents such as “Order”, “Despatch Advice”, and “Invoice” that are constructed from the UBL library components and can be used in generic procurement and transportation contexts.

A standard basis for XML business schemas provides the following advantages:

— Lower cost of integration, both among and within enterprises, through the reuse of common data structures.

— Lower cost of commercial software, because software written to process a given XML tag set is much easier to develop than software that can handle an unlimited number of tag sets.

— An easier learning curve, because users need master just a single library.

— Lower cost of entry and therefore quicker adoption by micro, small and medium-size enterprises (MSMEs).

— Standardized training, resulting in many skilled workers.

— A universally available pool of system integrators.

— Standardized, inexpensive data input and output tools.

— A standard target for inexpensive off-the-shelf business software.

UBL is designed to provide a universally understood and recognized syntax for legally binding business documents and to operate within a standard business framework such as ISO/IEC 15000 (ebXML) to provide a complete, standards-based infrastructure that can extend the benefits of existing EDI systems to businesses of all sizes. UBL is freely available to everyone without legal encumbrance or licensing fees.

UBL schemas are modular, reusable, and extensible in XML-aware ways. As an implementation of UN/CEFACT Core Components Technical Specification 2.01, the UBL Library is based on a conceptual model of information components known as Business Information Entities (BIEs). These components are assembled into specific document models such as Order and Invoice. These document models are then transformed in accordance with UBL Naming and Design Rules’ [28] use of the OASIS Business Document Naming and Design Rules [10] into W3C XSD schema syntax. This approach facilitates the creation of UBL-based document types beyond those specified in this release.

UBL can also be regarded as a generic Open-edi Configuration in the perspective of the Open-edi Reference Model (ISO/IEC 14662:2010). This is described in more detail in Annex H The Open-edi reference model perspective of UBL.

The intended primary audiences for this specification are:

— those who analyse and document business or processes or systems, assessing the business model or its integration with technology;

— those involved in the identification of business requirements for solutions to support the exchange of the digital business documents;

— those involved in the design, operation and implementation of software and services for the exchange of digital business documents; or

— those involved in the design, integration and operation of business applications dealing with digital documents.

Information Technology — Universal Business Language Version 2.2 (UBL v2.2)

1 Scope

ISO/IEC 19845 specifies the OASIS Universal Business Language (UBL) v2.2, which defines a generic XML interchange format for a selection of procurement and transportation business documents that can be restricted or extended to meet the requirements of particular industries. Specifically, UBL provides the following:

— A suite of structured business objects and their associated semantics expressed as reusable data components and common business documents.

— A library of XML schemas for reusable data components such as “Address”, “Item”, and “Payment”, the common data elements of everyday business documents.

— A set of XML schemas for common business documents such as “Order”, “Despatch Advice”, “Invoice”, “Waybill” and “Transportation Status” that are constructed from the UBL library components and can be used in generic procurement and transportation contexts.

ISO/IEC 19845 does not constrain the values used in documents, beyond the lexical constraints of value representation such as currency amounts, numbers or encoded binary information. Of note, values from coded domains, such as internationally- or privately-published code lists, are not constrained by the normative XML schemas. For the convenience of users, this specification does include a non-normative description of a second-pass value validation process and provides an optional suite of such value tests based on internationally-published code lists.

2 Normative References

1 Key words for use in RFCs to Indicate Requirement Levels

2 XML Advanced Electronic Signatures. ETSI TS 101 903 V1.4.1, June 2009.

3 Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000

4 XML-Signature Syntax and Processing. W3C Recommendation 12 February 2002

5 XML Schema Part 1: Structures. Second Edition. W3C Recommendation 28 October 2004

6 XML Schema Part 2: Datatypes. Second Edition. W3C Recommendation 28 October 2004

3 Terminology

3.1 Terms and Definitions

3.1.1
ASiC-S
Associated Signature Container (simple form). A standard container that associates a single data object with one or more detached signature(s) that apply to it. See [9].

3.1.2
Digital Signature
A value generated from the application of a private key to a message via a cryptographic algorithm such that it has the properties of integrity and message authentication and/or signer authentication. A signature may be (non-exclusively) described as detached, enveloping, or enveloped ([4], with modifications).

3.1.3
Document
A set of information components that are exchanged as part of a business transaction; for example, in placing an order.

3.1.4
Transform
The processing of data from its source to its derived form. Typical transforms include XML Canonicalization [31] and XSLT [35].

3.1.5
XSD schema
An XML document definition conforming to the W3C XML Schema language [5][6].

3.2 Other Terms and Definitions

The terms *Core Component (CC), Basic Core Component (BCC), Aggregate Core Component (ACC), Association Core Component (ASCC), Business Information Entity (BIE), Basic Business Information Entity (BBIE),* and *Aggregate Business Information Entity (ABIE)* are used in this specification with the meanings given in [12].

The terms *Object Class, Property Term, Representation Term,* and *Qualifier* are used in this specification with the meanings given in [22].

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY and OPTIONAL, when they appear in this document, are to be interpreted as described in [1].

4 Symbols and Abbreviated Terms

ABIE	Aggregate Business Information Entity
AdES	Advanced Electronic Signature
ASBIE	Association Business Information Entity
BBIE	Basic Business Information Entity
BIE	Business Information Entity
C14N	Canonicalization
CPFR	Collaborative Planning, Forecasting, and Replenishment [13]
DSig	Digital Signature
EDI	Electronic Data Interchange
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
NDR	Naming and Design Rules
QC	Qualified Certificate
QS	Qualified Signature
UML	Unified Modeling Language [29]
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
URI	Uniform Resource Identifier
XAdES	XML Advanced Electronic Signatures [2]
XML	Extensible Markup Language [3]
XMLDSig	XML Digital Signature [4]
XPath	The XML Path Language [32]
XSD	W3C XML Schema Language [5][6]
XSLT	Extensible Stylesheet Language Transformations (a transformation language) [34] [35]

5 UBL 2.2 Business Objects

5.1 Business Object Overview

The processes described in this section, and the business rules associated with them, define a context for the use of UBL 2.2 business documents. They are normative insofar as they provide semantics for the UBL document schemas, but they should not be construed as limiting the application of those schemas.

UBL 2.2 extends the generalized supply chain processes of UBL 2.0 (including the commercial collaborations of international trade) to include support for collaborative planning, forecasting, and replenishment; vendor managed inventory; utility billing; tendering; and intermodal freight management.

The following diagrams illustrate the business context use case covered by UBL 2.2.

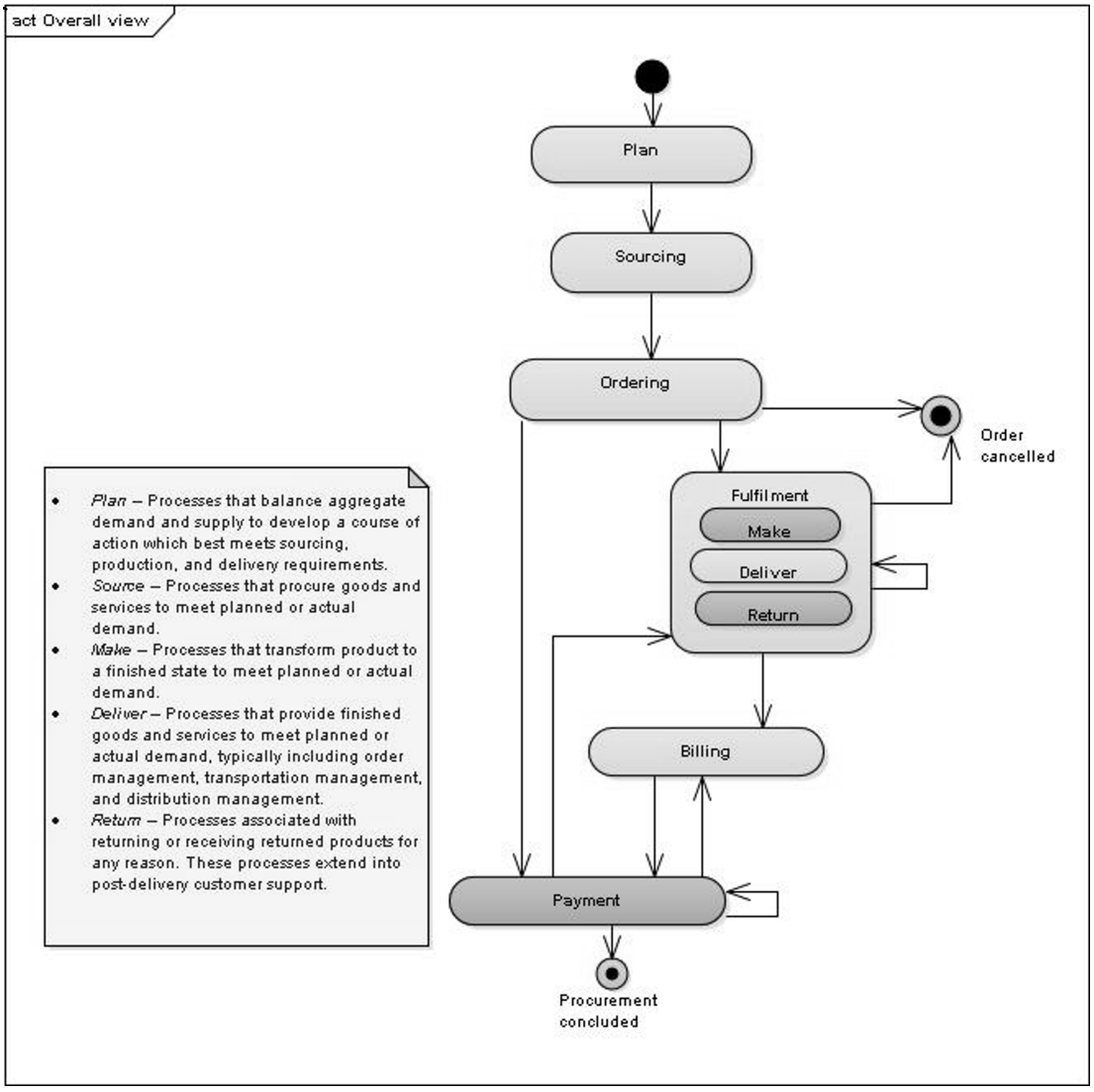

Figure 1 — UBL 2.2 Use Case Overall View

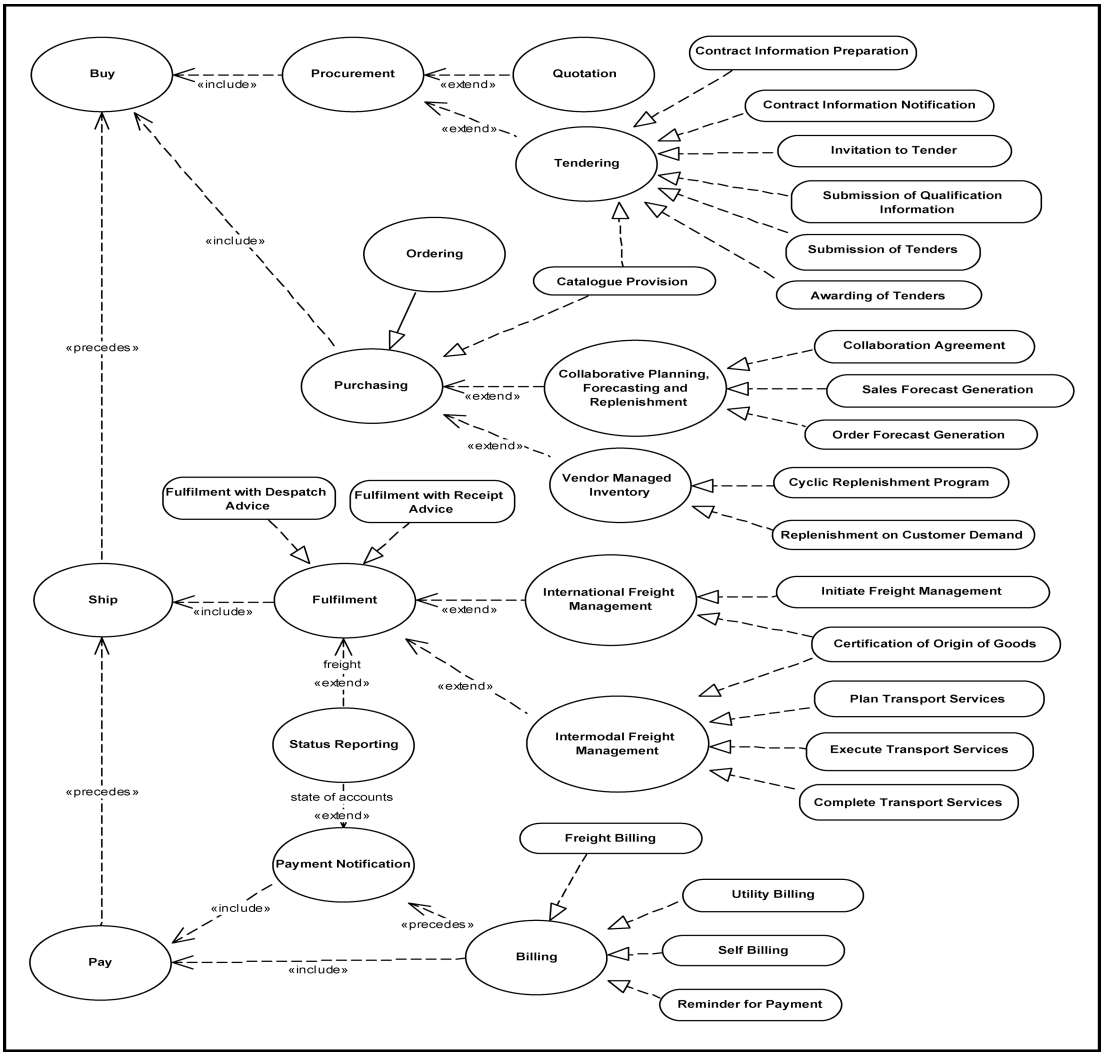

Figure 2 — UBL 2.2 Use Case Detail

The document types included in UBL 2.2 are listed in 6 UBL 2.2 Schemas. It is important to note that, as with previous UBL releases, the UBL 2.2 library is designed to support the construction of a wide variety of document types beyond those provided in the 2.2 package. It is expected that implementers will develop their own customized document types and components and that more UBL document types will be added as the library evolves.

For guidance in customizing UBL document types, see the UBL Guidelines for Customization [15].

For guidance in submitting recommended additions to and new UBL document types, see the UBL Maintenance Governance Procedures [21].

5.2 General Business Rules

5.2.1 General Business Rules Introduction

This section describes some of the requirements and general business rules that are assumed for collaborations and document exchanges using UBL 2.2.

5.2.2 Manifest Values

All information items in a UBL document are specified by the sender either as they are valued or as they are determined by some manner of a calculation model. For examples, an element may contain a fixed value, such as a name, or may contain a calculated value, such as one that is derived as the sum of other elements' values. The way a value is established or perhaps based upon a calculation model may or may not be documented by the sender. This imposes obligations on the sender when creating the UBL.

All fixed and calculated values must be manifest in the UBL instance. The receiver cannot presume to know that the sender has omitted an absent value as an assumption or as an indication of any kind that is pertinent to how the information is processed. Moreover, the sender cannot rely on the receiver deriving absent values from received values. The onus is on the sender to include all information, such as all pertinent indications and all relevant sums or calculations. The receiver need not make any assumptions nor perform any computations whatsoever when dealing with the sender's information.

An example receiver application is a print facility that can print any instance of a given UBL document type without having to perform any calculations nor need even know the underlying calculation model.

5.2.3 Items

— An item may be a product (goods) or a service

— Items may have multiple classifications

— A contract may influence prices of items

— An item may be part of another item

— An item may have a price per unit and an order unit

— An item may reference pictures and documents

— An item may have a validity period

— An item may refer to other relevant or necessary items

NOTE For a discussion of the difference between *item* and *line item* see 5.2.11 Item vs. Line Item.

5.2.4 Item Identification

One of the following identifiers may be used to identify each Item (for example, a product):

— Buyer's Item Identification, or

— Seller's Item Identification, or

— Manufacturer's Item Identification, or

— Catalogue Item Identification, or

— Item Identification according to a system promulgated by a standards body, industry group, or community of use.

The Item may be further distinguished by the specification of Measurement(s) or Physical Attribute(s). This enables specification of the following kinds of item:

— Item Requiring Description

  This is an item that is not identified by an unambiguous machine-processable identifier and requires additional descriptive information to precisely identify it.

— Customer Defined Item

  This is an item that the customer describes according to his need, and in the specification of which the customer may make some reference to comparable "standard" items.

— Item Requiring Measurements

  This is an item for which it is necessary to specify one or more measurements as part of the descriptive specification of the item.

5.2.5 Item Instances

Certain Items may be identified and ordered as individual, unique objects—for example, a specific car rather than a make and model of a car. This form of identification may also be needed for product tracing (e.g., perishable goods) or because of the nature of the commodity (e.g., used, collectible, specialized, or rare).

In data modeling terms, an Item Instance is an extension of an Item.

5.2.6 Item Pricing

For any given Item, price ranges by amount, quantity, location, etc., are specified by the Seller during the sourcing stage. They are not repeated back to the Seller during Ordering; only the active price is specified.

In some cases, the Buyer may not know the Item Price, in which case it is not specified. This makes a detailed response from the Seller necessary; see 5.3.3.4.4 Order Response.

5.2.7 Hazardous Items

Although ordered items may include Hazardous items, it is not necessary to specify information related to Hazardous status at the order stage. The Buyer may not be aware of the nature of the Item. Indication of the Hazardous nature of the Item, and any relevant information, would be indicated in the Despatch Advice and Transportation documents.

5.2.8 Parties

In UBL, a party is defined as an individual, a group, or a body having a role in a business function. Dependent on the business process, a Party may play various roles in the document exchange. For a list of UBL parties and their roles, see 5.4 Party Roles.

5.2.9 Multilingual Text

Some textual components, such as Notes and Description, may be specified in several languages. Each should be a separate occurrence of the component, using the language attribute to define its presentation. However, multiple occurrences of the same textual components should not be in the same language.

5.2.10 Taxation Rules

UBL does not provide documents for tax reporting purposes. Instead, it provides structures to support the information on which taxes are based. These aim to be generic and not based on any specific tax regime.

5.2.11 Item vs. Line Item

Many of the UBL document types employ the concept of a “line” inherited from traditional paper documents such as purchase orders and invoices. As in these older realizations, a “line” is a substantial data object with a number of sub-fields, typically including a short description, quantity, unit name, unit price, extension, and so on. Often in UBL these data structures include an element named `Item` that describes more fully the item of sale being ordered, invoiced, shipped, etc. `Item` in the line context always refers to the generic item of sale, not a unique, trackable, individual instance of such an item.

In the case of line structures such as `InvoiceLine` and `TenderLine`, the relationship between the line and the `Item` it contains is unproblematic, but a person unfamiliar with traditional usage may easily be confused by the line element called `LineItem`. In traditional business processes, “line item” is a common name for the entire line structure in a purchase order or invoice, *not just the item of sale contained in the line.* Thus, despite the name, a `LineItem` is not an `Item` but rather a complex data structure that *contains* an `Item` along with quantity, price, and so on.

5.2.12 Shipment vs. Consignment

References to “shipment” and “consignment” appear in a number of places in the UBL data model relating to the transport of goods. For IT specialists unfamiliar with the way these terms are used in international trade, the structural relationships between the two can be puzzling. For example, a close look at the data model shows that shipments can comprise multiple consignments and consignments can comprise multiple shipments. This is not a design flaw but rather a reflection of the possible real-world relationships between the two concepts.

Shipment and *consignment* actually refer to two different ways of looking at the same (possibly very complex) situation. From the physical or logistical point of view, a *consignment* is the transportation of an identifiable collection of goods items from one party (the consignor) to another (the consignee) via one or more modes of transport. From the contractual or logical point of view, a *shipment* is the *contractual arrangement* whereby an identifiable collection of goods items is to be transported from one party (the shipper) to another party (the recipient). In UBL, the party originating the shipment is usually a supplier, and the party receiving the shipment is usually a buyer.

In the simplest fulfilment scenario, these distinctions are almost invisible; see Figure 3 below (used, like the subsequent three, by permission of Document Engineering Services). In this case, the supplier of the contracted shipment is the consignor of the physical goods, and the buyer is the consignee.

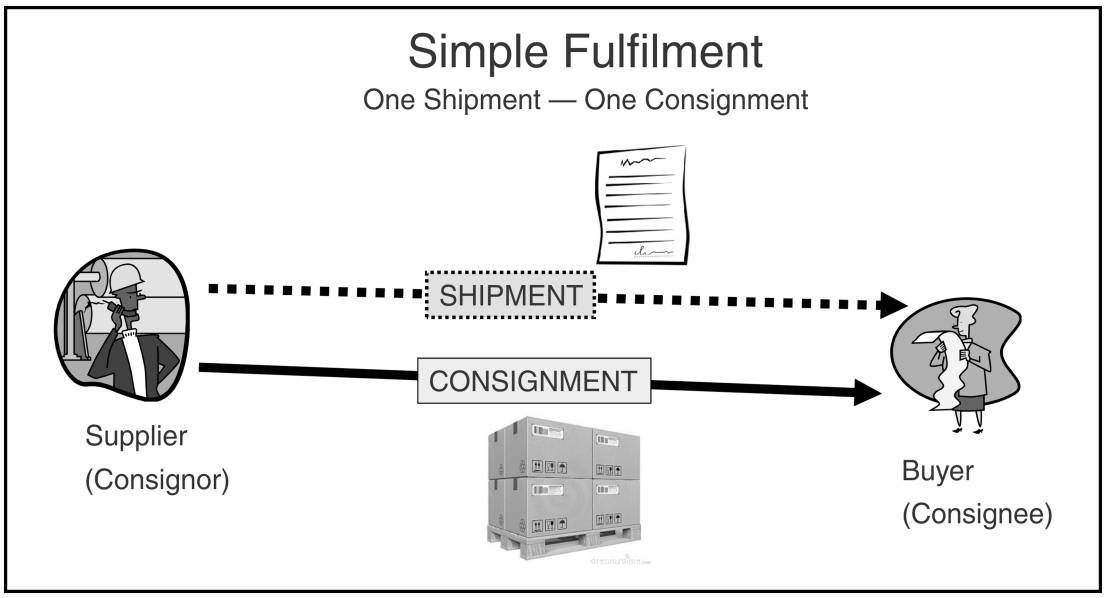

Figure 3 — Simple Fulfilment

Often, however, a single contractual shipment is split up into separate physical consignments that may be received on separate schedules, as shown in Figure 4. The shipper may use multiple carriers, or the shipment may be so large that it must be transported in multiple vessels, becoming in effect multiple consignments. It is therefore often necessary for the UBL description of a shipment to contain descriptions of the consignments into which the goods have been divided.

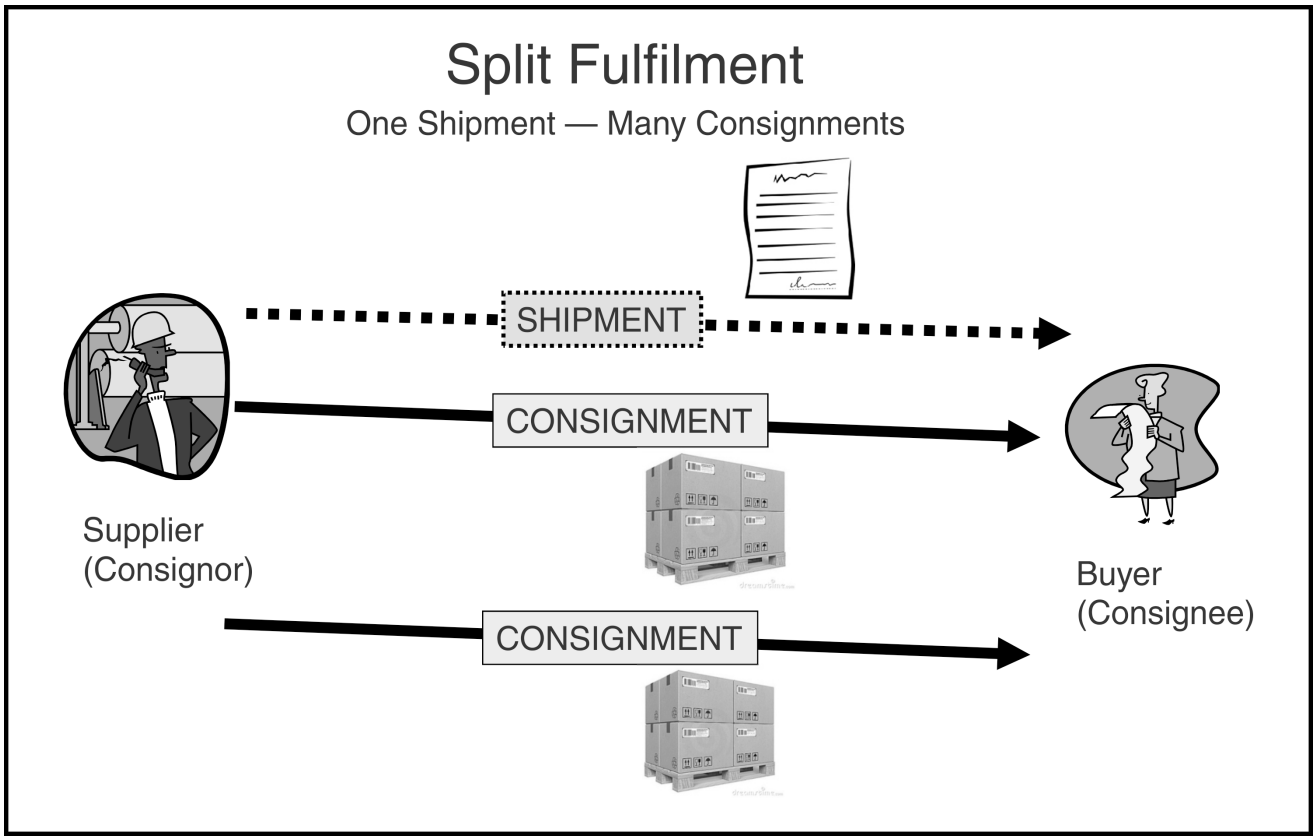

Figure 4 — Split Fulfilment

So far, the shipper (here a supplier) remains the only consignor and the recipient (here the buyer) the only consignee. But sometimes the division of a shipment into consignments takes place "behind the scenes" through the involvement of a freight forwarder, who becomes both a second consignee and a second consignor (Figure 5). The "shipment" in this case is the entire end-to-end organization of the transport of goods on behalf of the shipper.

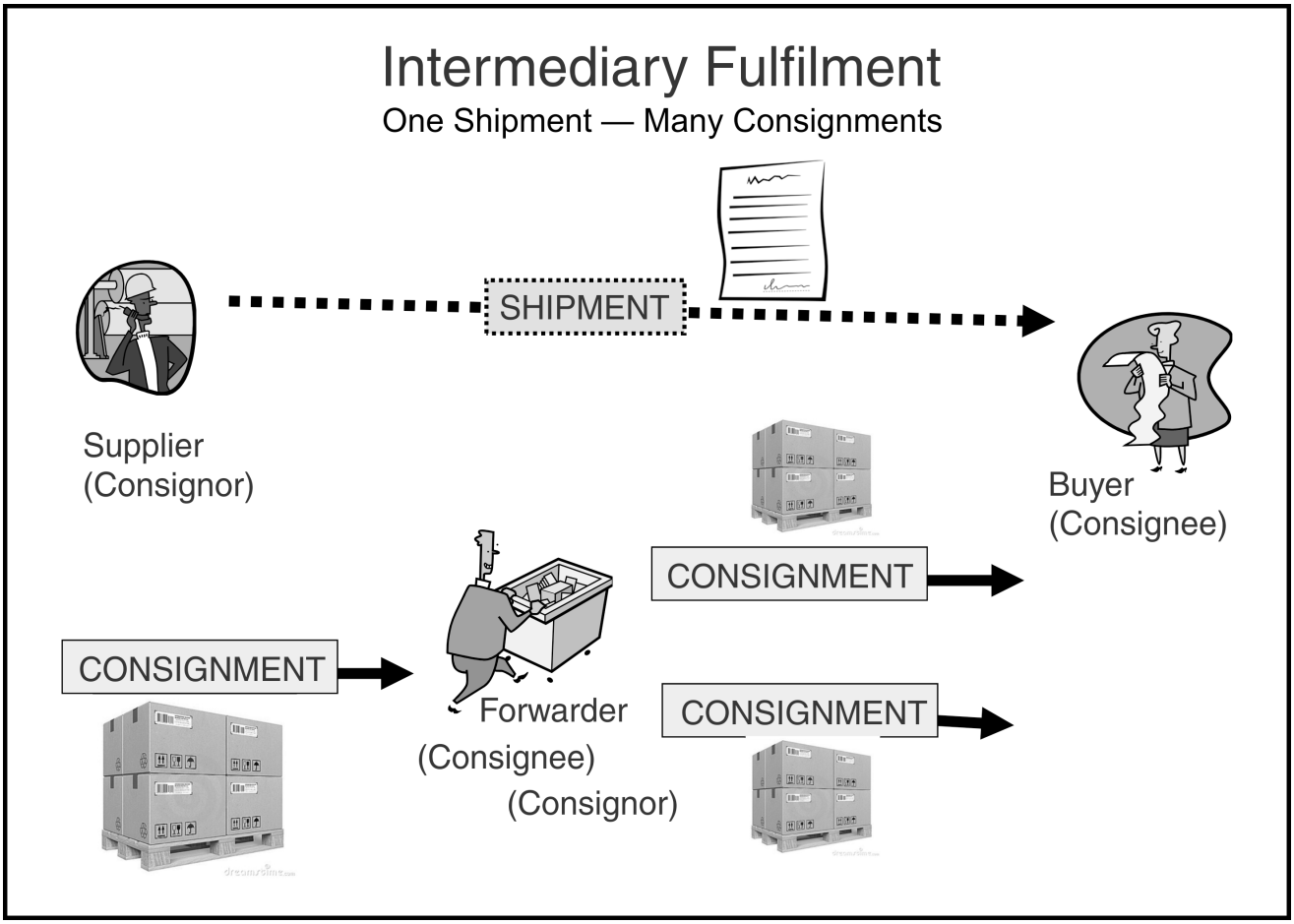

Figure 5 — Intermediary Fulfilment

Another layer of complexity is introduced when pieces of different, possibly unrelated shipments are consolidated into a single consignment to make the physical process more efficient (to share space in the same shipping container, for example, which optimizes transport by ensuring that the container is fully loaded and also provides a more competitive tariff). In Figure 6, goods from two completely unrelated business transactions between two buyers and their suppliers — two different shipments — are consolidated by a freight forwarder into a single consignment for part of their journey and then separated again by another freight forwarder farther on. This requires the UBL description of the consignment to contain descriptions of the shipments participating in the consolidation. Note that the transaction between the two freight forwarders is itself a shipment (a *consolidated shipment*), and its data structure must be able to describe the two shipments it is covering (Supplier A to Buyer A and Supplier B to Buyer B) so that the receiving forwarder knows how to de-consolidate the consignment.

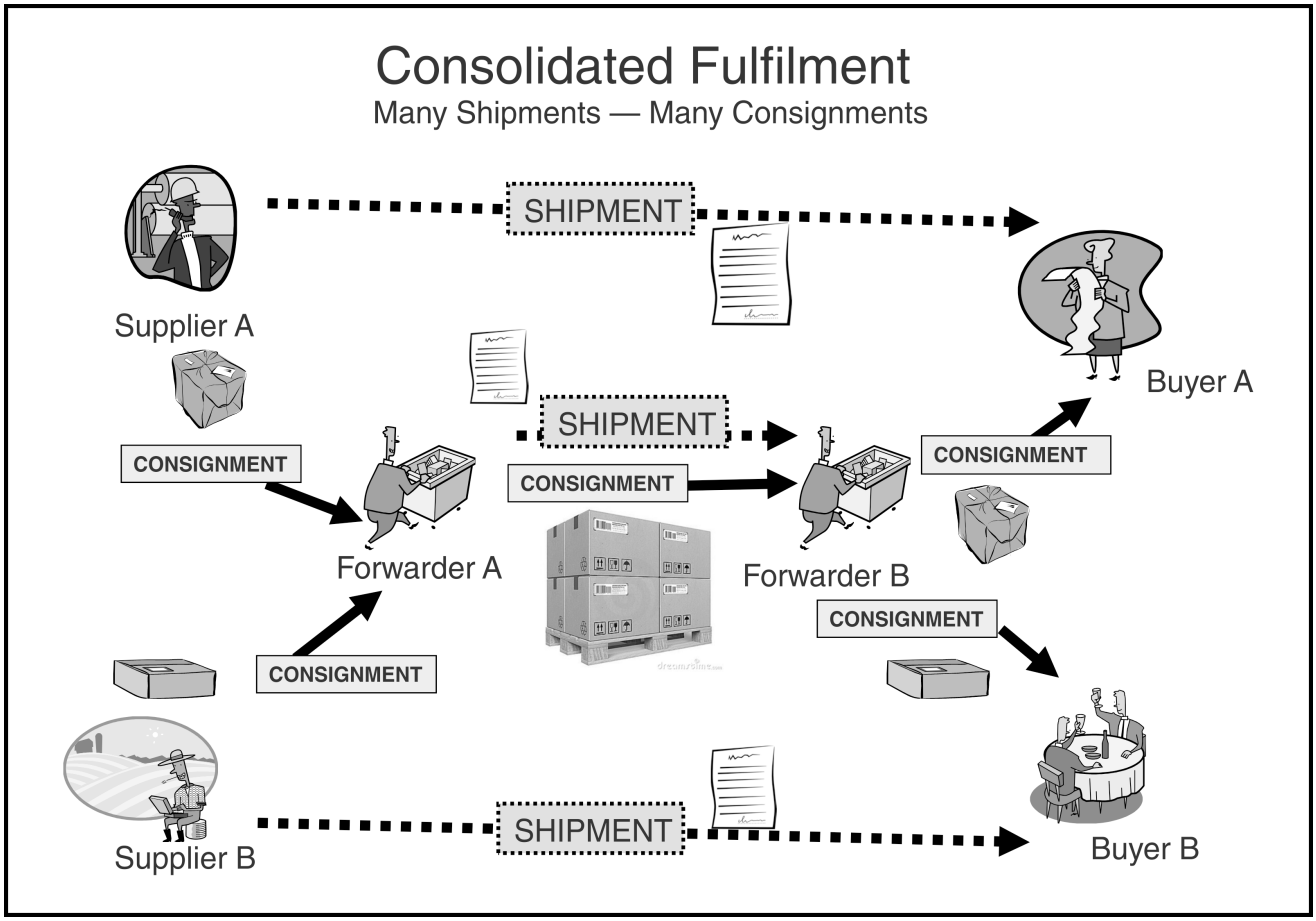

Figure 6 — Consolidated Fulfilment

Note that the word “consignment” in the context of transportation has a meaning different from that of “consignment” in sales and vendor-managed inventory (5.3.3.5 Vendor Managed Inventory).

5.2.13 Transport vs. Transportation

The terms “transport” and “transportation” both appear many times in the UBL data model. There is no semantic difference between these terms as used in UBL; in the context of freight management, they mean exactly the same thing: the conveyance of goods or persons.

“Transportation” is the oldest of the two forms, the noun “transportation” first appearing in written English about 70 years earlier than the noun “transport”. UBL 2.0 adopted “transportation” as the preferred form in terms such as “transportation service” and “transportation status”, but in the process of developing UBL, which features greatly expanded data representation capabilities for multimodal freight management, it became clear that “transport” is the form to be preferred, both because it is shorter and because it is the more commonly used of the two in international contexts. The decision to adopt “transport” for new usages while preserving backward compatibility with UBL 2.0 by retaining “transportation” in data items from the earlier release has resulted in the mixed terminology seen here.

5.2.14 Transport Events

There are two methods of capturing Transport Event information: at the Consignment level and at the Shipment Stage level.

A Consignment may pass through several shipment stages in its lifetime, for maritime shipments this would typically be pre-carriage, main carriage and on-carriage stages. Each of these stages has events such as pickups and deliveries. In these scenarios the Shipment Stage is the appropriate structure for containing the Transport Event information.

But it is also possible for the information to be a snapshot of the status of a Consignment (for example where the consignee and consignor are not aware of these stages). This view of the Consignment is as one set of Transport Events. In these scenarios the Consignment is the appropriate structure for holding the Transport Event information.

5.2.15 Financial Information

UBL has been enhanced to support the financial information required for downstream processing of Invoices within financial services. By aligning information models business vocabularies such as UBL for eBusiness and ISO 20022 for eFinance enable Straight Through Processing (STP) and paperless trading along the entire Financial Supply Chain. For example, the UBL Invoice and Remittance Advice can be used together with financial messages to ensure end-to-end transport of reconciliation identifiers (invoicing party references). In particular, UBL provides a solution for advanced external remittance, where the UBL Remittance Advice is used to transmit the details of complex remittance information associated with the payment initiation process (see ISO 20022 guides for details).

UBL is also designed to support basic trade financing practices (invoice financing, factoring, pre-shipment/order financing, Letter of Credit, etc.).

5.2.16 Indirect Taxes

The structure and semantics of UBL with respect to taxation information have been aligned with the OASIS Indirect Tax Reference Model Version 2.0 produced by the OASIS Tax XML TC supported by the OECD. The purpose of this reference model is to present a model of the tax related information contained within the messages exchanged between the participants involved in a business transaction, the primary purpose of which is not tax-related, but which may be subject to the imposition of an indirect tax. This model is intended to serve as a reference for any effort to analyze the related messages (documents) of an implementation to verify that the indirect tax implications are adequately addressed, and as input to any effort to define message-oriented specifications involving indirect taxation. It is based on a three party scenario, where parties in a commercial business process can conduct their transactions and provide taxation, customs or independently auditable information when required.

5.3 Supply Chain Business Processes

5.3.1 Supply Chain Overview

Following from UBL 2.1, the UBL 2.2 library and documents support an increased range of different business processes. See B.5 Minor Revision: UBL 2.2 for a detailed summary of the changes to the library and documents. The UBL business processes now supported can be categorized as follows (those with document type additions in 2.2 are shown in italicized boldface):

- 5.3.2 Plan
  - 5.3.2.1 Collaborative Planning, Forecasting, and Replenishment
    - 5.3.2.1.1 Collaborative Planning, Forecasting, and Replenishment Introduction
    - 5.3.2.1.2 Collaboration Agreement and Joint Business Planning
    - 5.3.2.1.3 Sales Forecast Generation and Exception Handling
    - 5.3.2.1.4 Order Forecast Generation and Exception Handling
- 5.3.3 Source (procurement)
  - 5.3.3.1 Tendering (pre-award)
    - 5.3.3.1.1 Tendering Introduction
    - 5.3.3.1.2 Contract Information Preparation
    - 5.3.3.1.3 Contract Information Notification
    - 5.3.3.1.4 Invitation to Tender

5.3.8.4 Digital Agreement

5.3.2 Plan

5.3.2.1 Collaborative Planning, Forecasting, and Replenishment

5.3.2.1.1 Collaborative Planning, Forecasting, and Replenishment Introduction

The VICS Collaborative Planning, Forecasting, and Replenishment (CPFR®) guidelines [13] formalize the processes by which two trading partners agree upon a joint plan to forecast and monitor sales through replenishment and to recognize and respond to any exceptions.

In the UBL context of use, these CPFR processes between the retailer and the manufacturer have been extended to cover the planning process between other parties such as the manufacturer and the supplier. These binary collaboration definitions are the template guidelines for implementers to build their own collaboration process based on their supply chain topology and requirements.

As shown in Figure 2-2 of [13], the seller and the buyer engage in three main activities in order to improve the overall performance of the supply chain:

1) **Planning** establishes the ground rules for the collaborative relationship. Trading partners exchange information about their corporate strategies and business plans in order to collaborate in the development of a Joint Business Plan. The Joint Business Plan identifies the significant events that affect supply and demand in the planning period, such as promotions, inventory policy changes, store openings/closings, and product introductions.

2) The **Forecasting** phase involves the development of a shared plan based on consumer demand. Estimation of consumer demand at the point of sale is called sales forecasting, and future product ordering based on the sales forecast is referred to as order forecast.

3) The **Replenishment** phase involves order generation, which transitions forecasts to firm demand, and order fulfilment, the process of producing, shipping, delivering, and stocking products for consumer purchase. Note: This phase may be implemented using other UBL processes.

A fourth collaborative activity, **Analysis,** involves monitoring the execution of activities for exceptions that are identified during the strategy and planning phase. Calculation of key performance metrics and plan adjustments for improving results also take place in Analysis. This activity is represented in the CPFR diagram by the arrows labeled “Exception Triggers” and the process called “Resolve/Collaborate on Exception Items” in the Forecasting phase.

While these collaboration activities are presented in logical order, most companies are involved in all of them at any moment in time. There is no predefined sequence of steps. Execution issues can impact strategy, and analysis can lead to adjustments in forecasts.

5.3.2.1.2 Collaboration Agreement and Joint Business Planning

The Collaboration Arrangement is the preparatory step that defines the scope of the project, assigns roles, establishes procedures for data interchange, and issues identification and resolution. The following actions are performed through meetings and agreements:

— Receive and review background information from the sales organization or buyers

— Identify the product categories that should be included in the initial scope

— Define Collaboration Objectives

— Define specific metrics that reflect the objectives

— Determine the Event collaboration cycle

— Determine the times of the review meetings to discuss the results

— Document the data sources that are essential for a successful event collaboration process, and

— Document additional information that can be used in the event analysis.

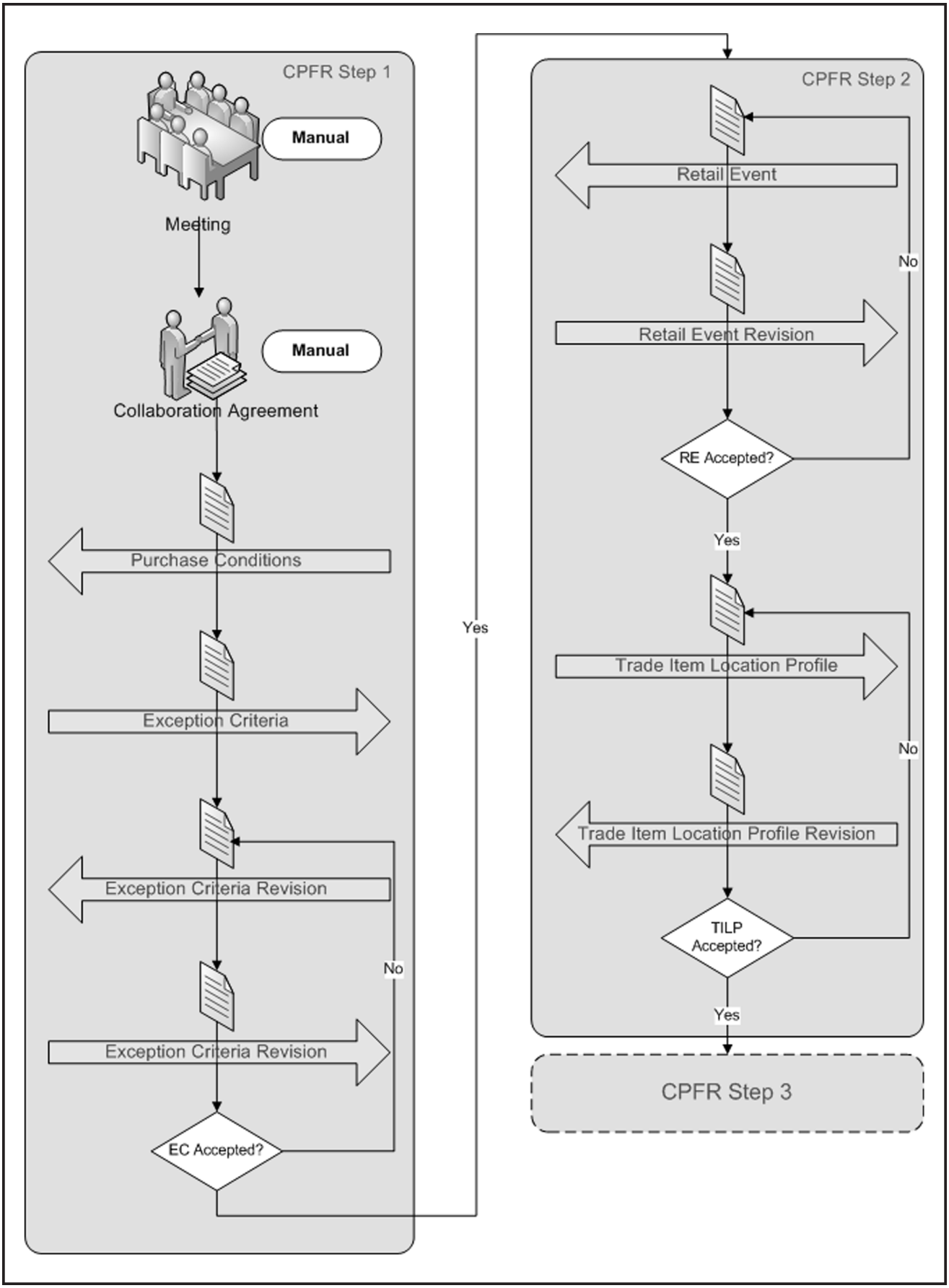

Figure 7 — CPFR Steps 1 and 2

The first step of the CPFR Process continues with the exchange of messages containing purchase conditions. (UBL does not standardize the format of such messages.) Afterwards, for determining the exception criteria that should be monitored and handled during the execution, Exception Criteria messages are exchanged. Exchange of revised Exception Criteria messages continues until the criteria are accepted by both sides.

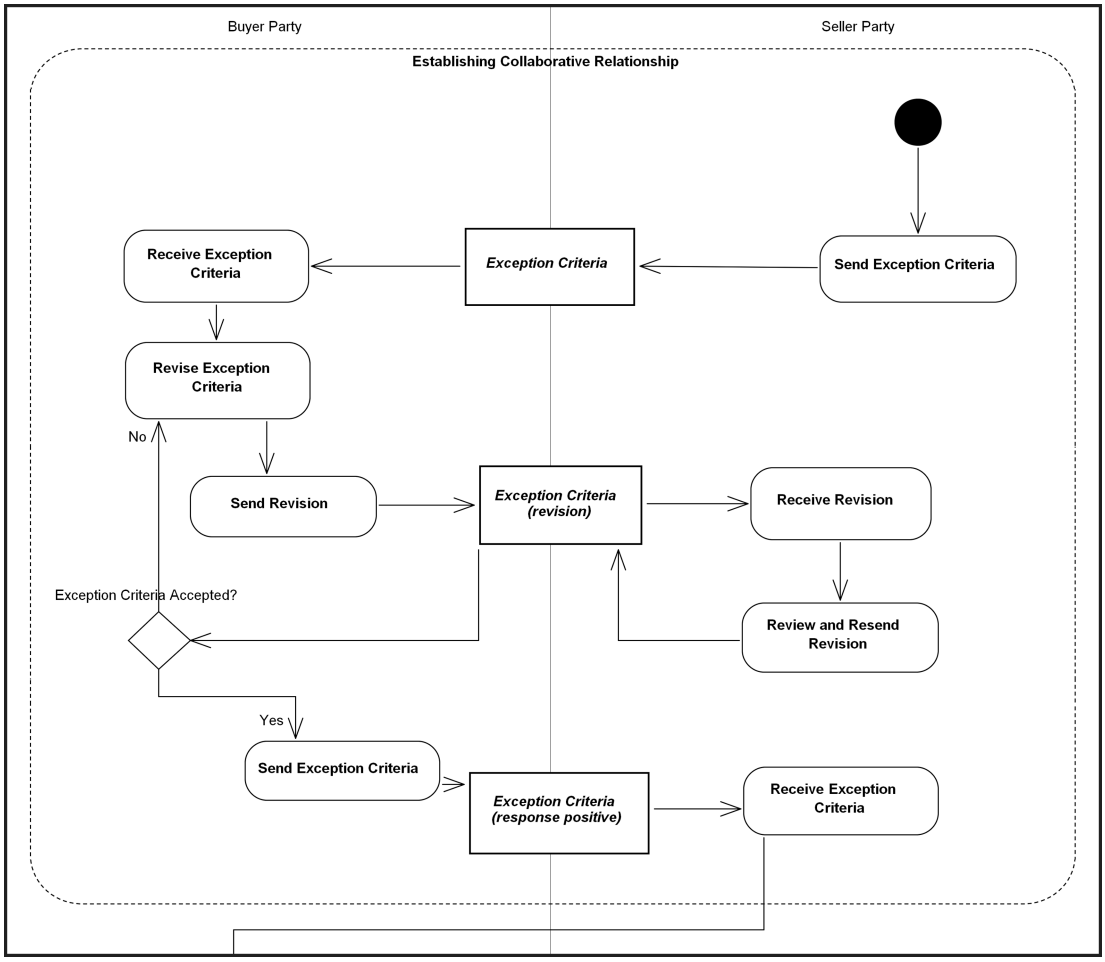

Figure 8 — Establish Collaborative Relationships

In CPFR Step 2 (the Joint Business Planning phase) there are two messages that should be exchanged and agreed upon: Retail Event and Trade Item Location Profile. Revisions are exchanged until an agreement is achieved.

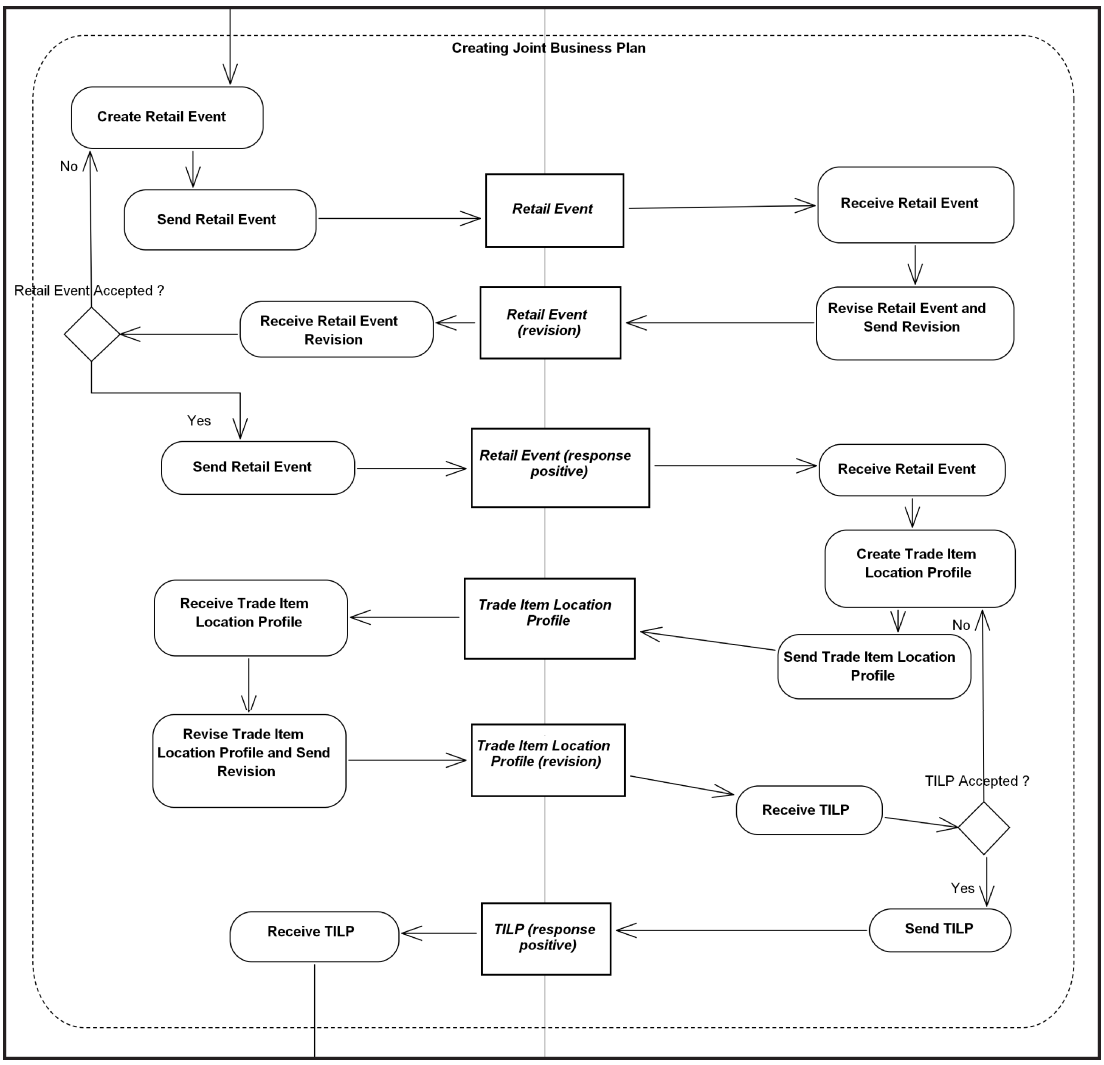

Figure 9 — Create Joint Business Plan

5.3.2.1.3 Sales Forecast Generation and Exception Handling

CPFR Step 2 helps the buyer and seller agree to the event details and calendar that meet their joint business and collaboration objectives. The objective of the event calendar is to ensure that events are planned to achieve the optimal results and to enable both parties to plan the execution of the event more accurately, from the preparation of advertising and displays to the production and delivery of the promotional stock.

In CPFR Step 3, the Sales Forecast is generated. Following Option A, Conventional Order Management, from the CPFR implementation scenarios (see [14], Table 3), the responsible partner for the generation of Sales Forecast is the Seller. Having Event Calendar information and the Delivery Plan already in their system, there are two more kinds of information that the Seller needs for an effective Sales Forecast: POS Data and DC Data. As shown in Figure 10 and Figure 11, both of these pieces of information are sent within a Product Activity message. This time there is no revision of the messages because these messages contain statistical and historical information collected previously by the Buyer.

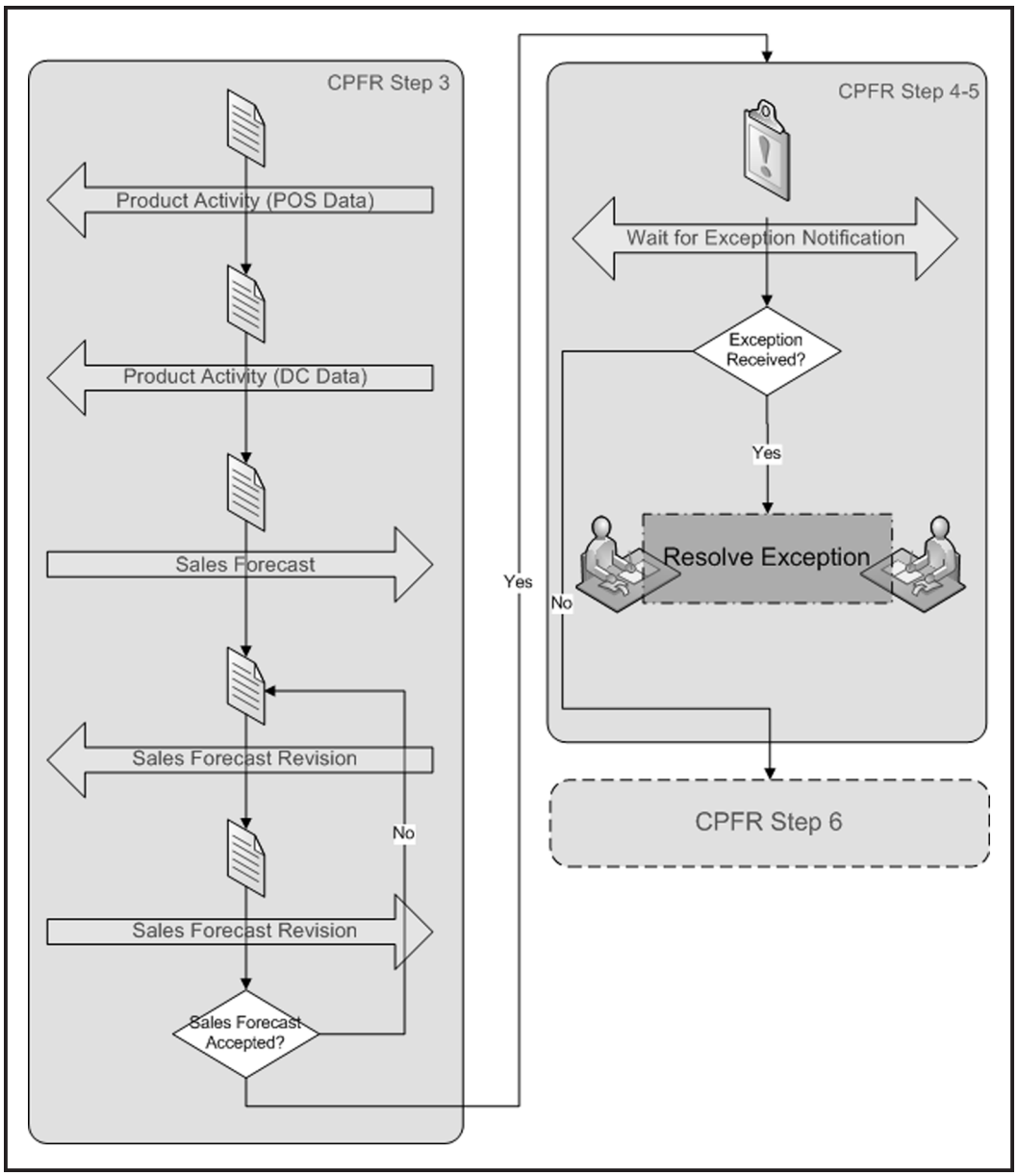

Figure 10 — CPFR Steps 3, 4, and 5

Based on the event details (dates, products, tactics, etc.) and using the available data source(s), a volume estimate/forecast is created for each product/store combination included in the scope of the event by the Seller. During the calculation, sales forecasting algorithms make use of the coefficients for causal factors based on the event history. Once the Sales Forecast suggestion is generated and sent to the Buyer, the Buyer revises it and might recommend some changes on the Forecast. The Forecast Revision message exchange continues until the forecast is agreed by both sides.

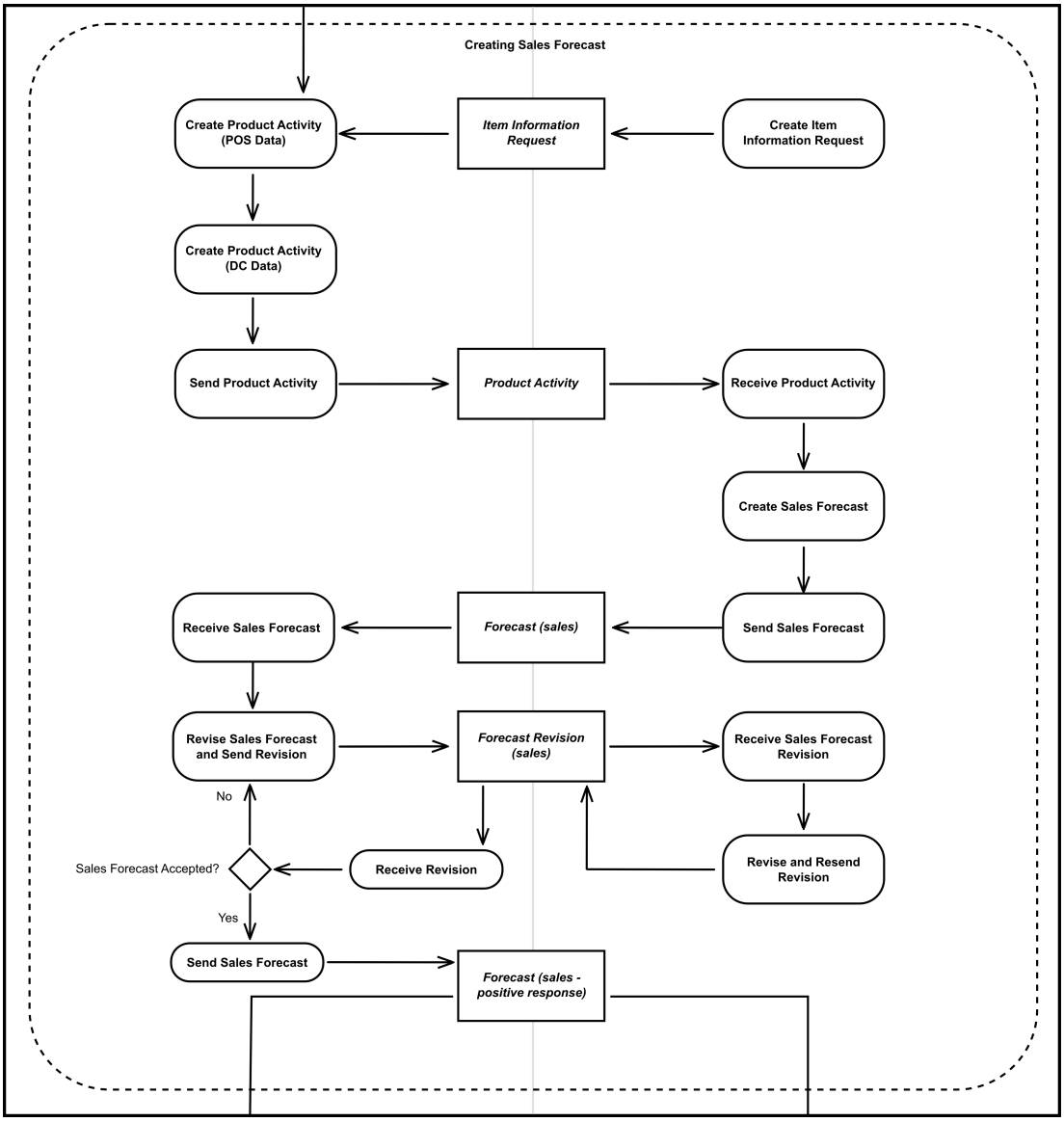

Figure 11 — Create Sales Forecast

In many cases some time may elapse between Sales Forecast Generation and Order Generation. During this period, both sides observe changes to the conditions. If one of the partners detects an exception invalidating the exception criteria defined in CPFR Step 1, it sends an Exception Notification message to the other party. Exceptional circumstances that may be communicated between trading partners include deviations between planned impacts (either between buyer and seller, or between subsequent generations of planned impacts from the same trading partner), as well as deviations between planned and actual impacts. It should be noted that both sides might detect an exception, and therefore both sides should be capable of sending and receiving exceptions. Of course, for specific implementations if the collaborating parties want to change this behaviour, they can customize the process so that one partner will be responsible for the generation of the Exception Notifications.

CPFR Step 4 is solely composed of the exception generation and receiving activity. CPFR Step 5, on the other hand, is the resolution of the Exceptions.

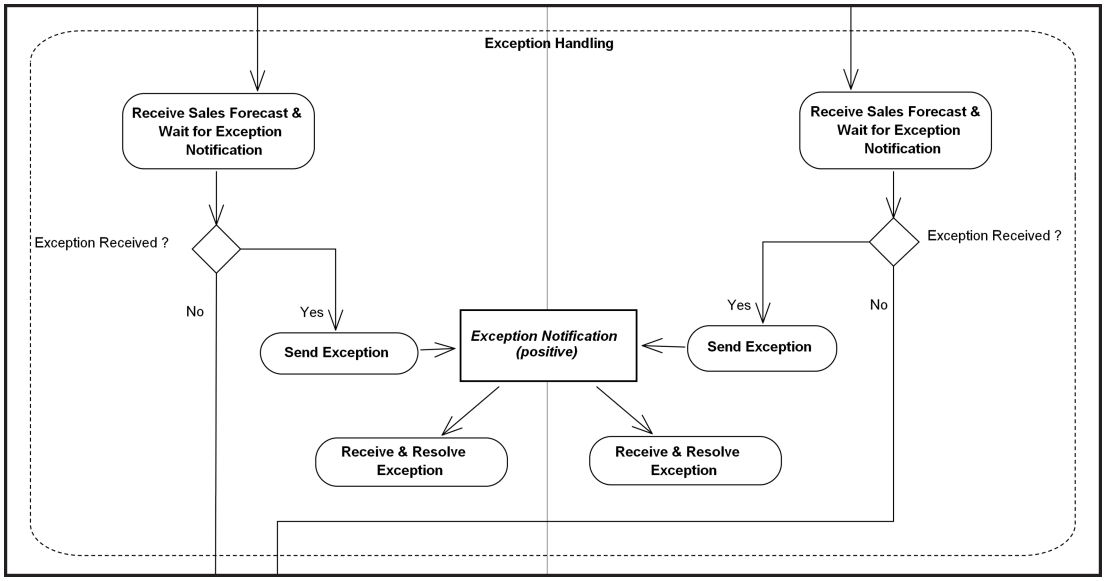

Figure 12 — Exception Handling

If there is no Exception Notification Message within the defined period, the process continues with Order Forecast Generation (CPFR Step 6).

5.3.2.1.4 Order Forecast Generation and Exception Handling

In the supply chain process, it is important for sales forecasts that are created to be converted into the shipment (order) forecasts that can then be used in the production planning processes at the manufacturing locations and be incorporated into the ordering processes at the retailer. As shown in Figure 13, the responsibility for creating Order Forecast belongs to the Seller per Option A of the CPFR implementation scenarios (see [14], Table 3). Sales forecasts can be transformed into order forecasts by incorporating inventory status information, possible retail event plans, and current point of sale data. Therefore, Buyer sends the updated versions of the Retail Event, Inventory Status, and POS Data to the Seller.

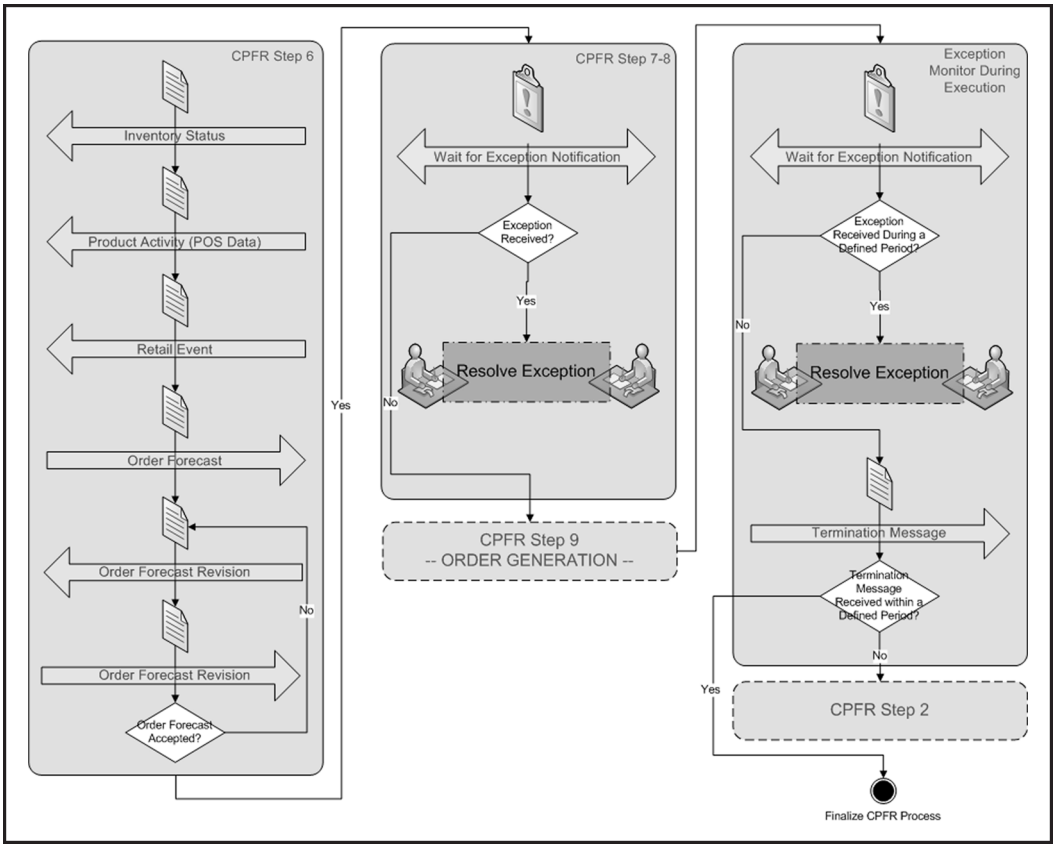

Figure 13 — CPFR Steps 6, 7, 8 and 9

After the Seller creates the Order Forecast using the obtained data, it sends the forecast to the Buyer. The Buyer checks the order forecast and sends back a revision document which includes update requests if necessary. The exchange of Order Forecast Revisions continues until there are no further update requests and the Order Forecast is agreed by both sides. Document types used in this process are Retail Event, Product Activity, Forecast, and Forecast Revision.

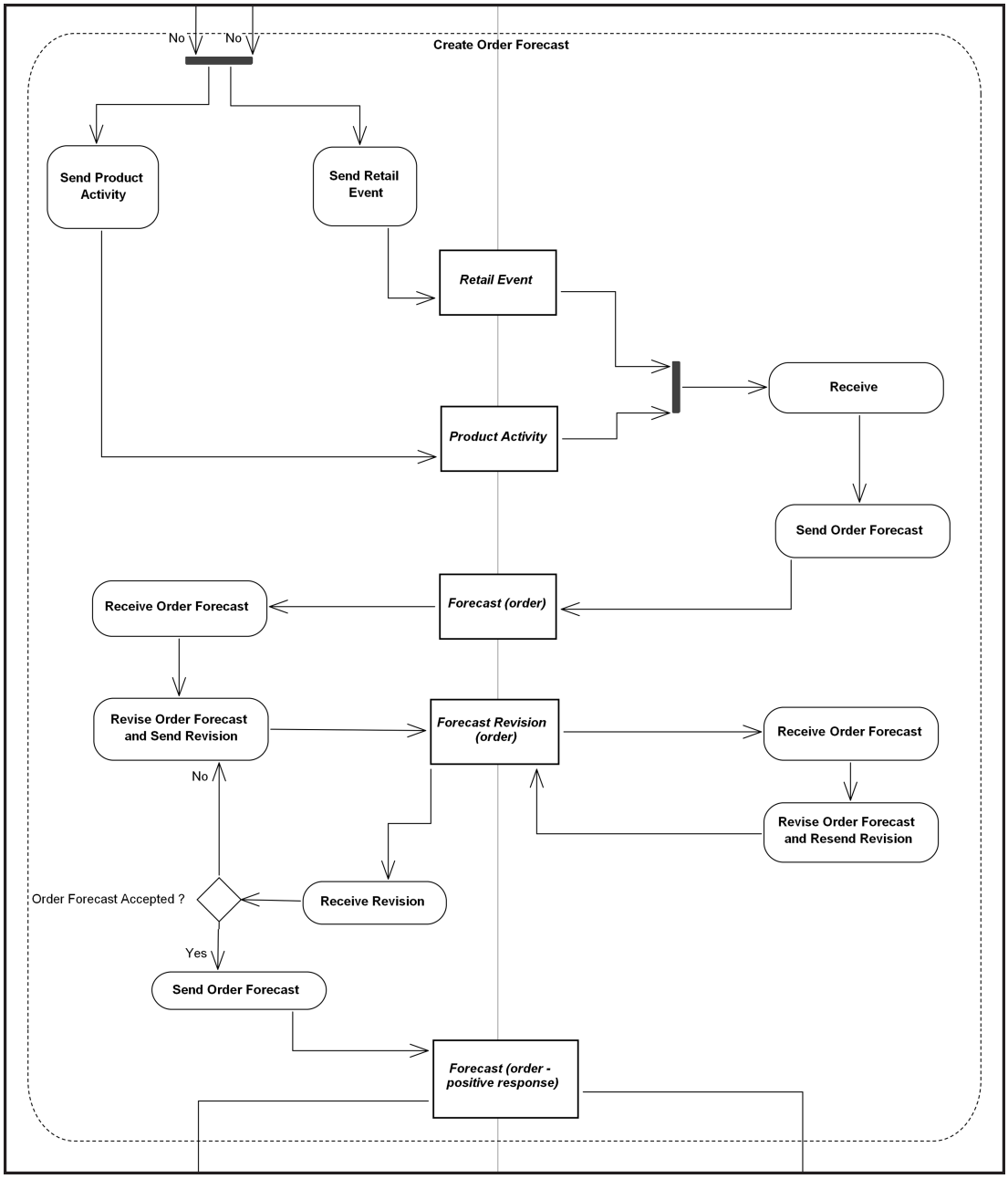

Figure 14 — Create Order Forecast

After the Order Forecast is frozen, the process continues with the exception detection activity (CPFR Step 7). The exception detection process that follows Order Forecast is similar to process described earlier for exception detection following Sales Forecast (see 5.3.2.1.3 Sales Forecast Generation and Exception Handling). The only difference between the Order Forecast and Sales Forecast exceptions is the content of the exceptions.

CPFR Step 8, Order Forecast Exception Resolution activity, is handled similarly to Sales Forecast Exception Resolution.

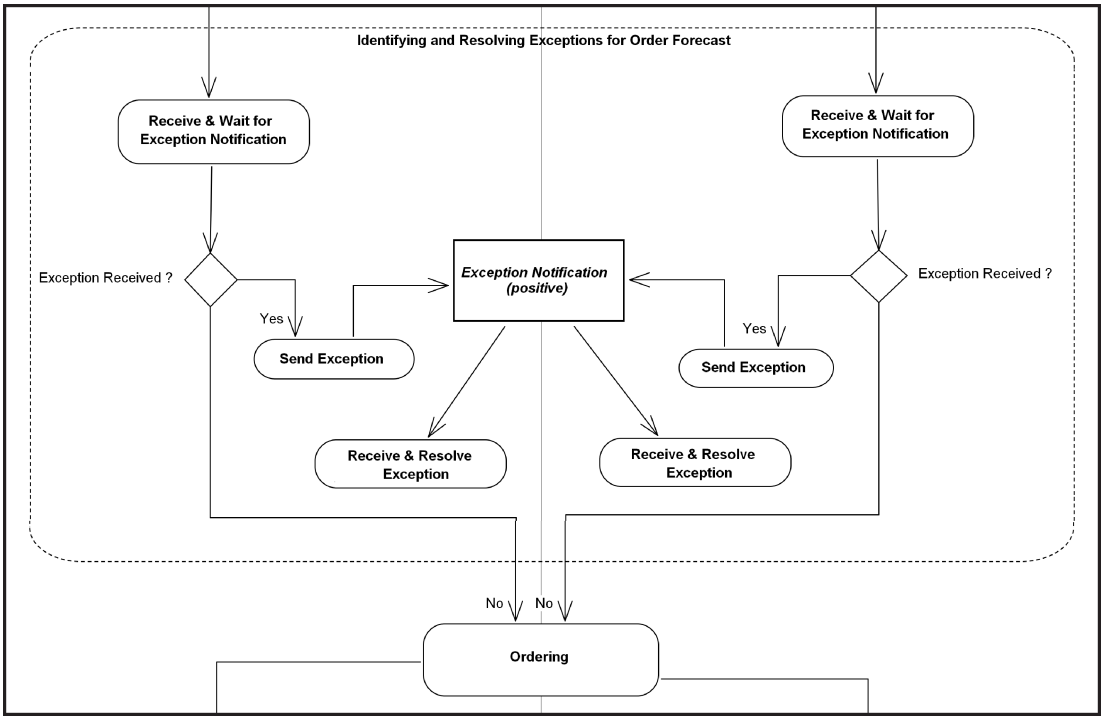

Figure 15 — Identifying and Resolving Exceptions for Order Forecast

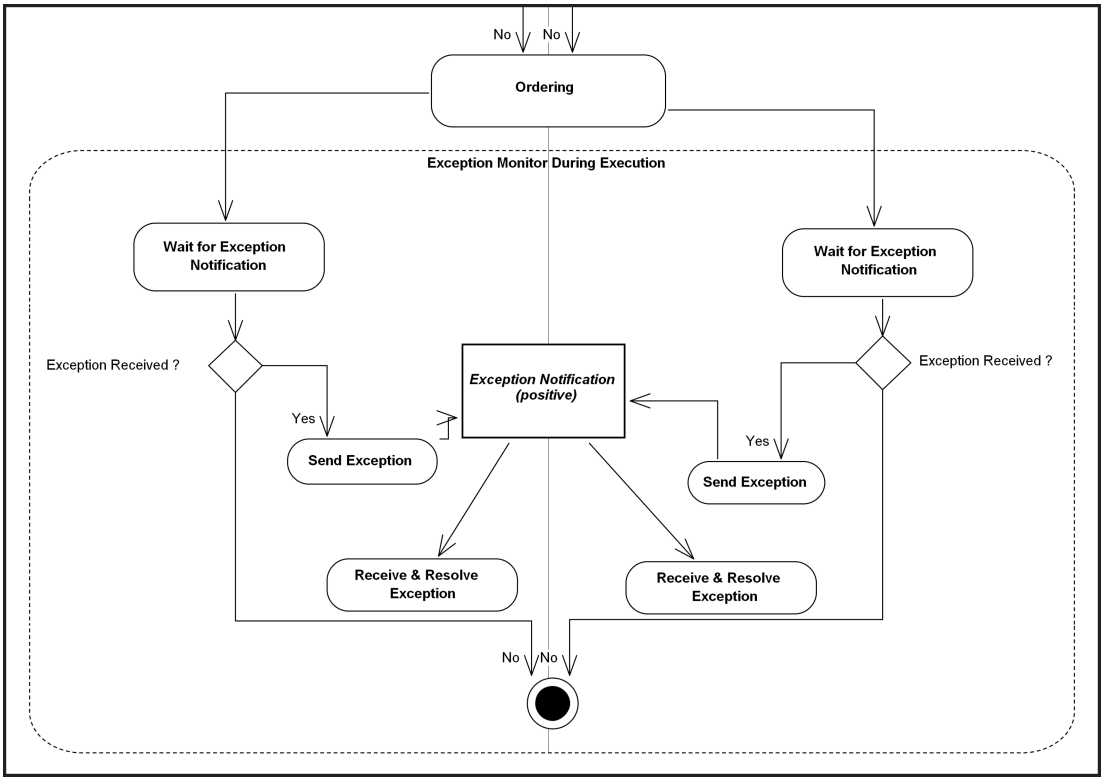

Figure 16 — Exception Monitor During Execution

If there is no exception during a period of time, the process continues with the Order Generation Step.

From the technical point of view, the exception monitoring and its resolution are exactly same as in the case of Order Forecast Exception Handling and Sales Forecast Exception Handling. The difference is in the content of the exceptions. The actual events and orders are compared to the Forecasted Sales and Forecasted Orders. When there is a situation violating the normal

exception criteria, one of the sides might generate an exception notification. Besides comparison of forecasts, other information gathered during the execution is observed (e.g., event dates, POS data, etc.). The resolution of the exceptions is the same as the process carried out for Sales Forecast Exception resolution.

5.3.3 Source (procurement)

5.3.3.1 Tendering (pre-award)

5.3.3.1.1 Tendering Introduction

Tendering is the case where a contracting authority (the Originator) initiates a procurement project to buy goods, services, or works during a specified period, as shown in the following diagram.

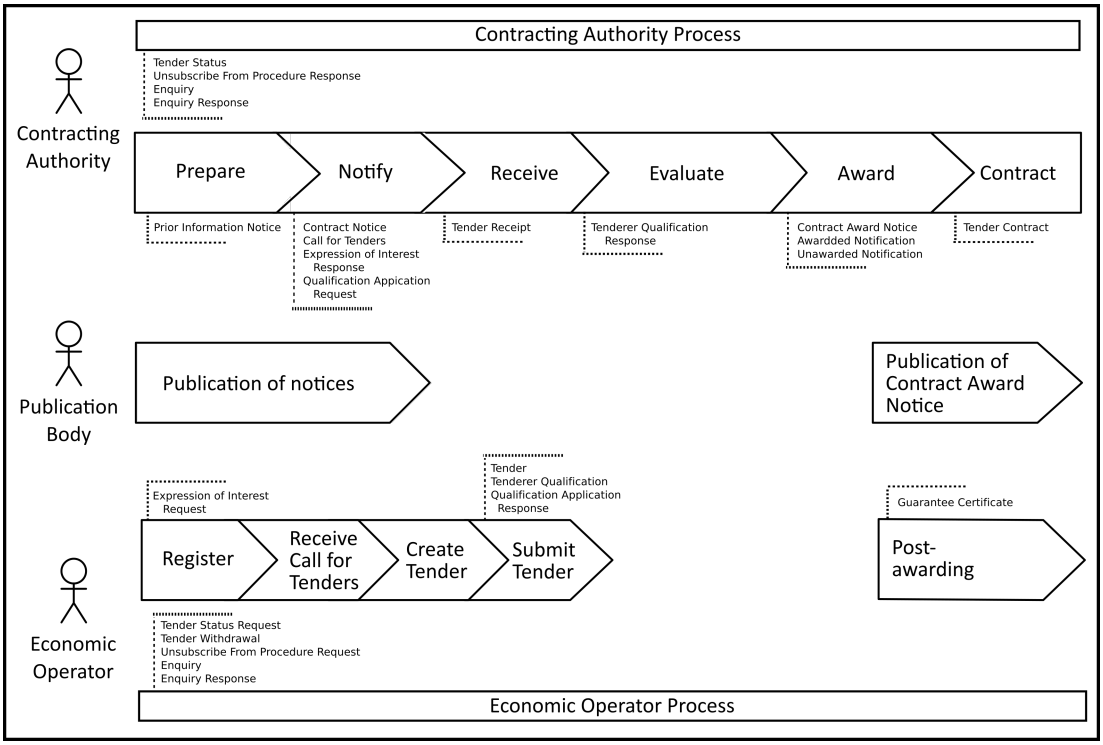

Figure 17 — The Tendering Process

A similar but less formally defined process than tendering is quotation (see 5.3.3.3 Quotation).

5.3.3.1.2 Contract Information Preparation

The Tendering process optionally begins with publication of a Prior Information Notice prepared by a Contracting Authority to *declare the intention* to buy goods, services, or works during a specified period. The purpose of this step (if implemented) is to reduce preparation time when an actual Contract Notice is published (see 5.3.3.1.3 Contract Information Notification).

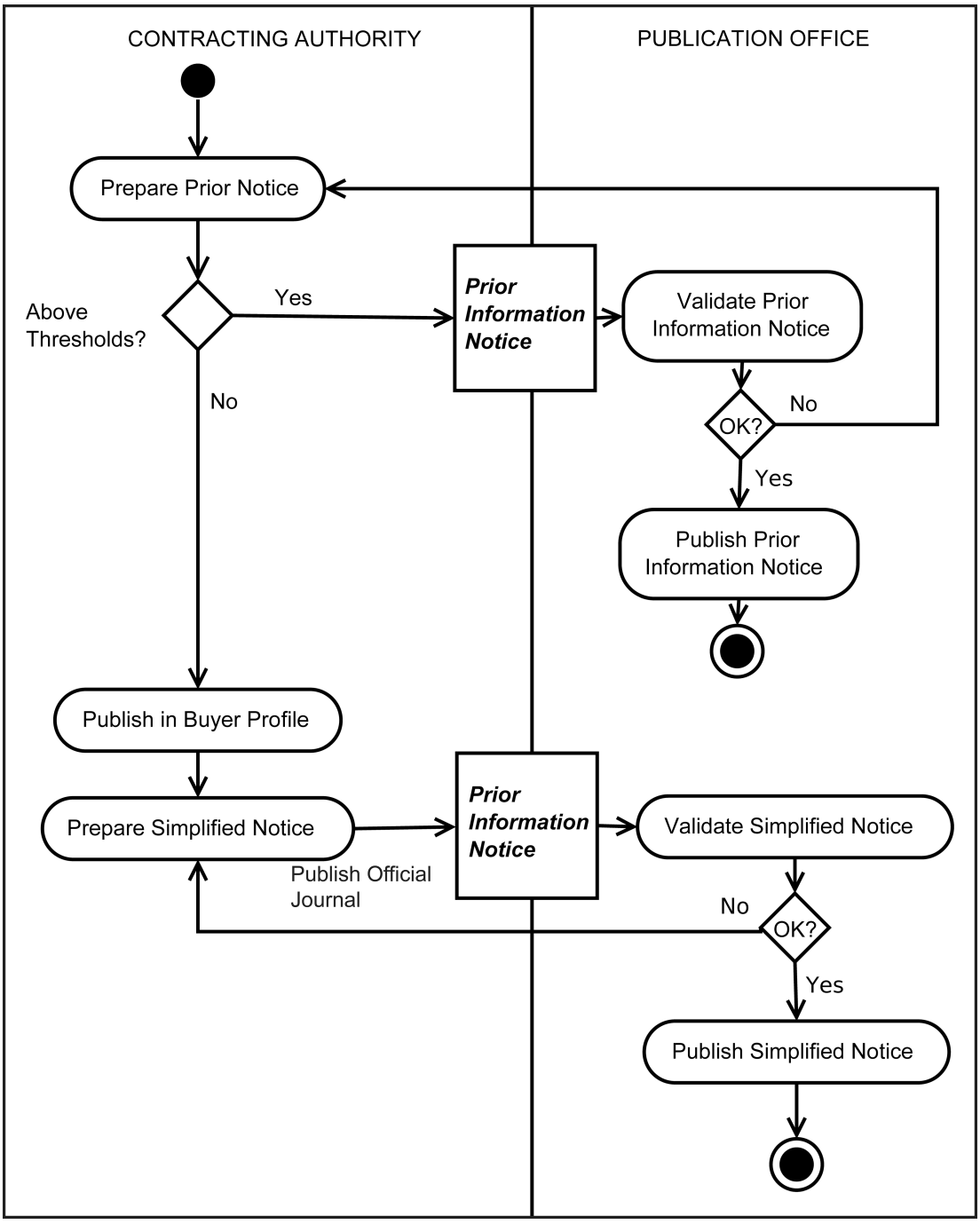

Figure 18 — Contract Information Preparation

5.3.3.1.3 Contract Information Notification

The process of Notification includes the publication by the Contracting Authority of a Contract Notice to *announce* the project to buy goods, services, or works. The details shown here are specific to the EU, which requires contracts over a certain amount (Harmonized contracts) to be published in the Official Journal of the EU. Other tendering contexts will differ in their publication requirements.

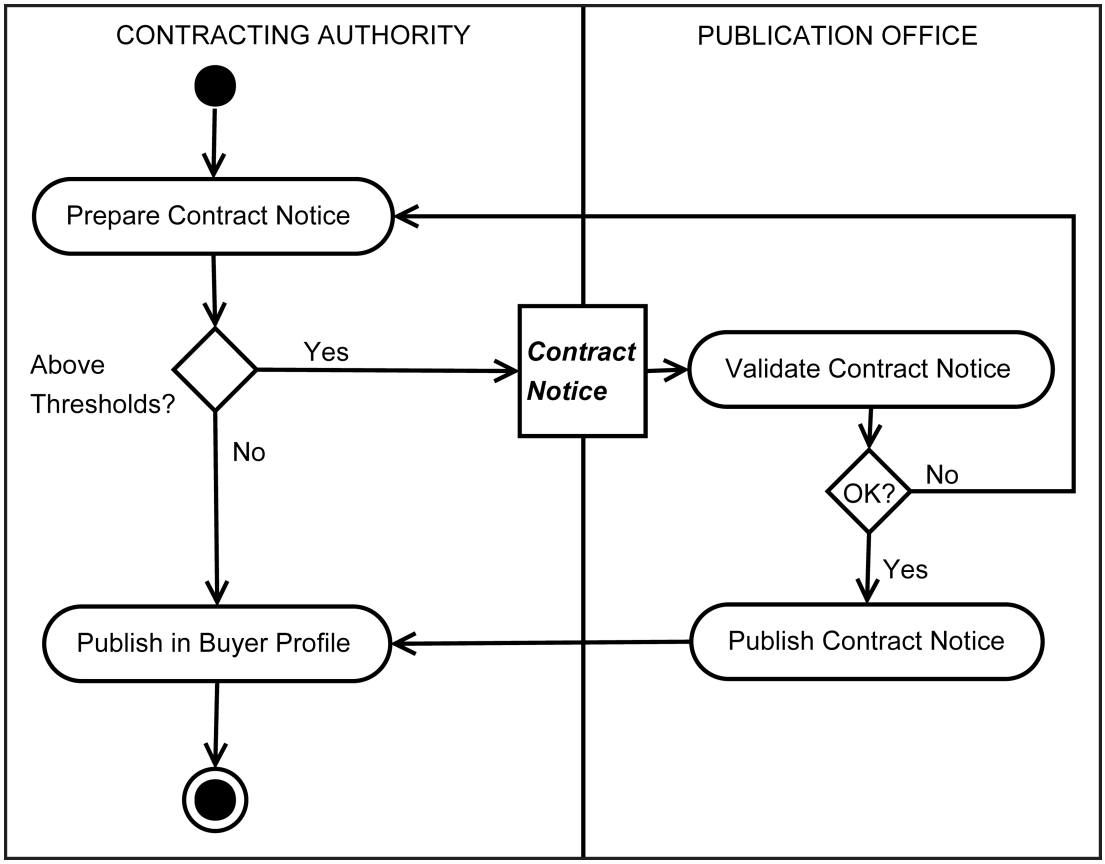

Figure 19 — Contract Information Notification

5.3.3.1.4 Invitation to Tender

In some procedures, the Contracting Authority invites economic operators to participate in a contest by sending them an invitation to tender using a Call For Tenders to *define* the procurement project to buy goods, services, or works during a specified period. The Call for Tenders may be sent jointly with an unstructured letter of invitation to tender.

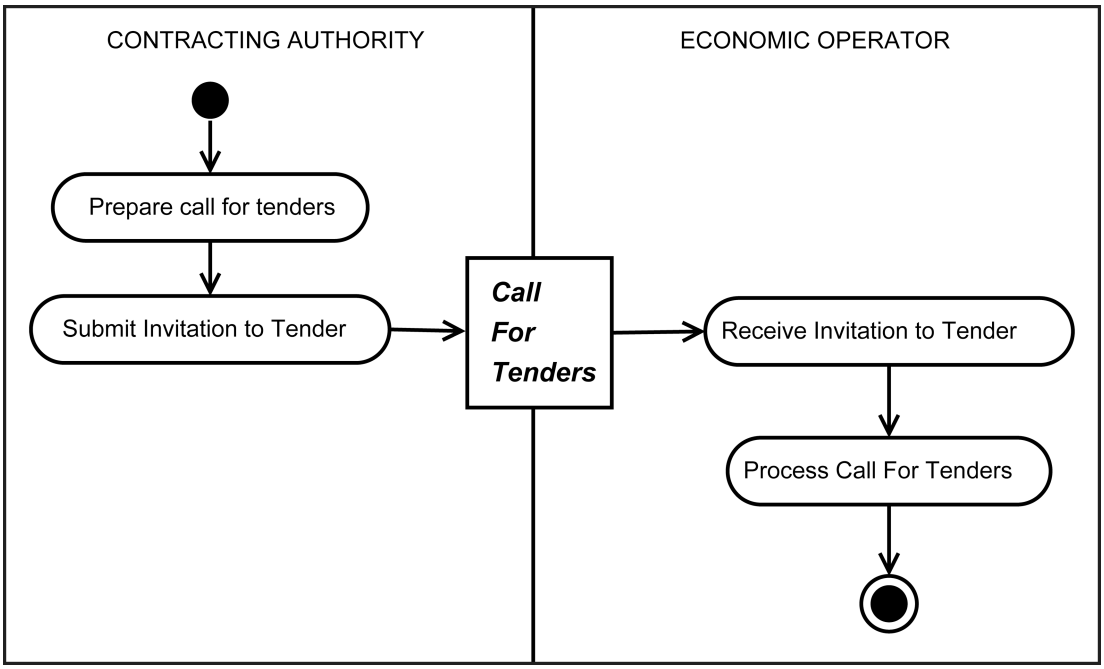

Figure 20 — Invitation to Tender

5.3.3.1.5 Expression of Interest

An economic operator expresses interest in a tendering process by submitting an Expression of Interest. The Contracting Authority replies with an Expression of Interest Conformation to confirm the economic operator will receive any modification of the terms and documents related with that tendering process.

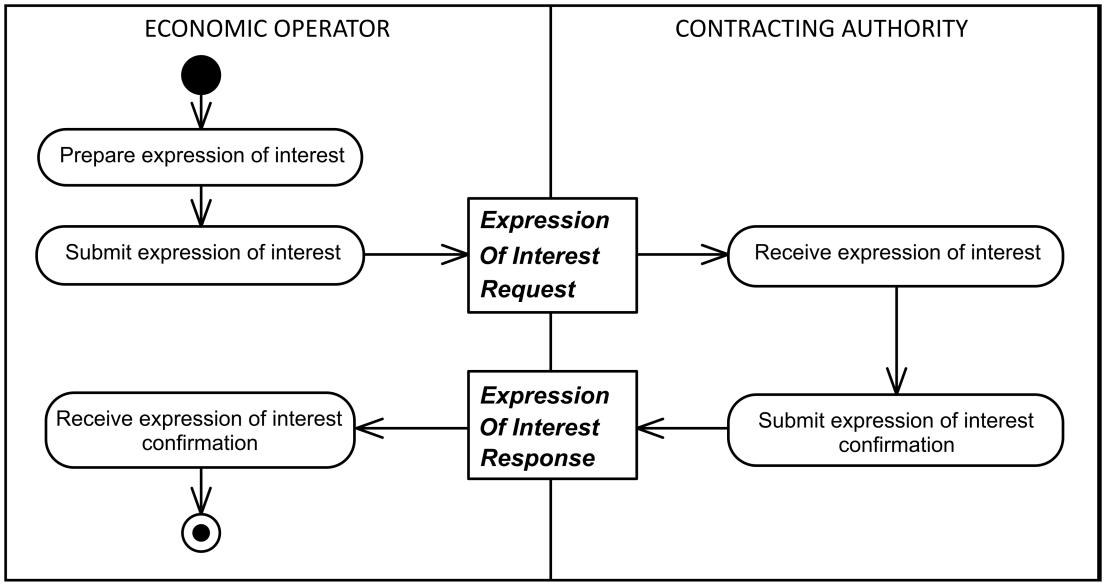

Figure 21 — Expression of Interest

5.3.3.1.6 Unsubscribe from Procedure

An economic operator requests to be unsubscribed from a tendering process by submitting an Unsubscribe From Procedure. The Contracting Authority replies with an Unsubscribe From Procedure Conformation to confirm the economic operator will be removed from the list of interested economic operators and will not receive any modification of the terms and documents related with that tendering process.

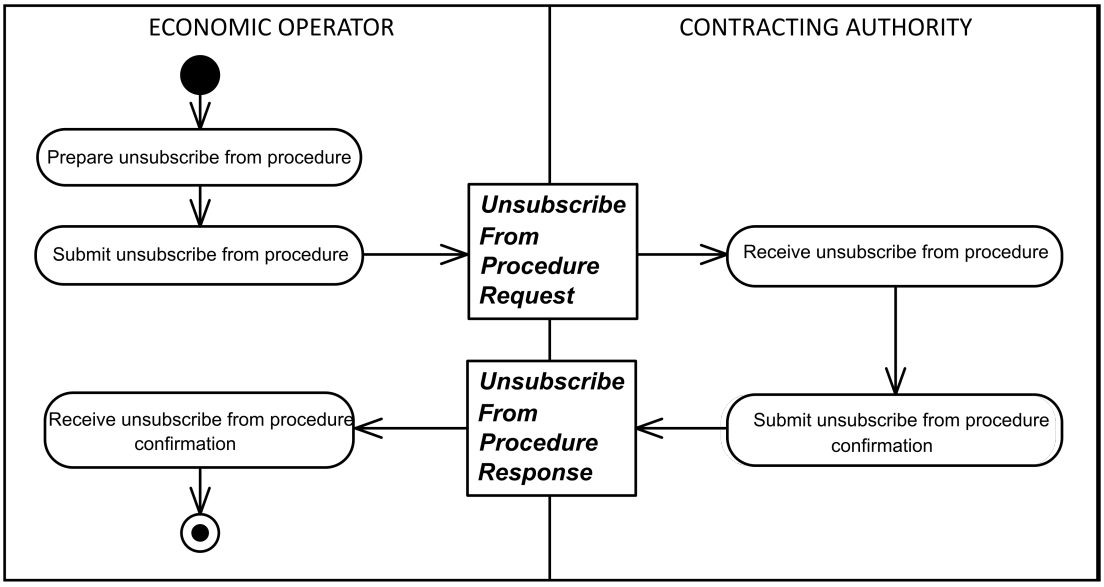

Figure 22 — Unsubscribe from Procedure

5.3.3.1.7 Submission of Qualification Information

The economic operator sends a Tenderer Qualification to the Contracting Authority to *define its own situation or status* relating to the requirements of the Contracting Authority for a specific tendering process. The Contracting Authority uses the Tenderer Qualification Response to notify the Tenderer of its *admission to or exclusion from the tendering process.*

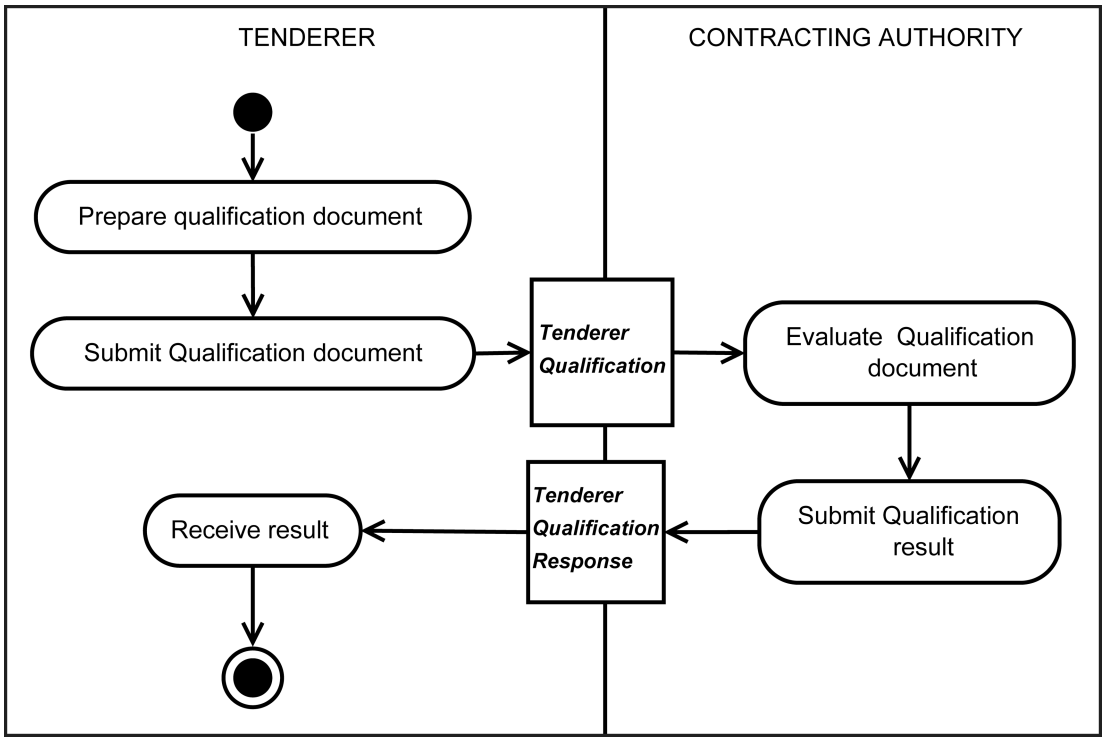

Figure 23 — Submission of Qualification Information

5.3.3.1.8 Qualification Application

A contracting authority makes a description of the required qualification application request (In Europe: ESPD Request) to an Economic Operator (the tenderer). The Economic Operator (the tenderer) makes a description of the required application qualification response (In Europe: ESPD Response) to a Contracting Authority in order to become eligible to participate in the tendering process.

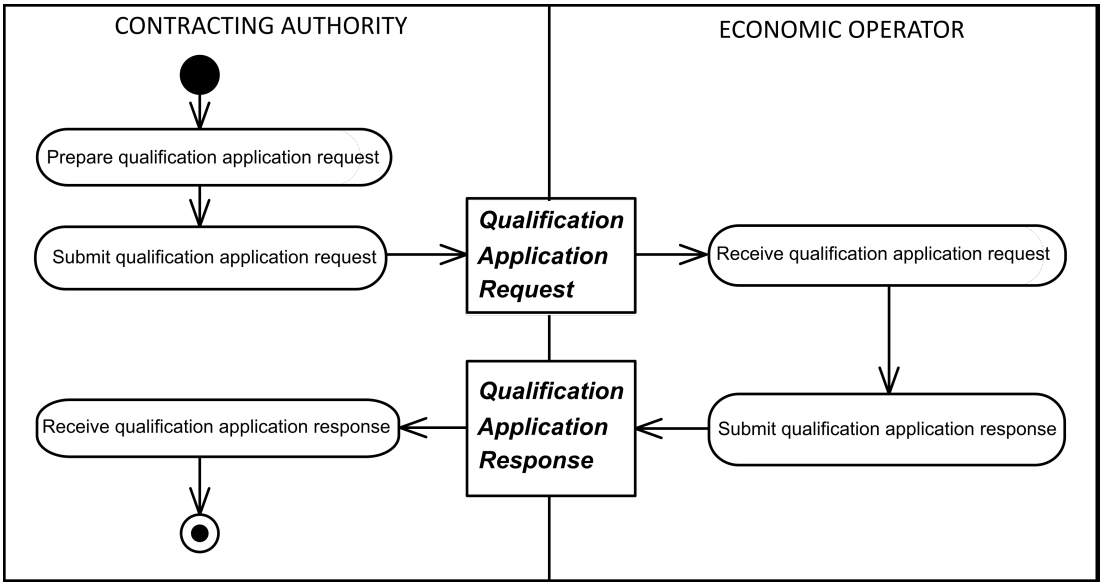

Figure 24 — Qualification of Interest

5.3.3.1.9 Enquiry

A requester sends a question to a responder using an Enquiry document and the responder replies with a Response document.

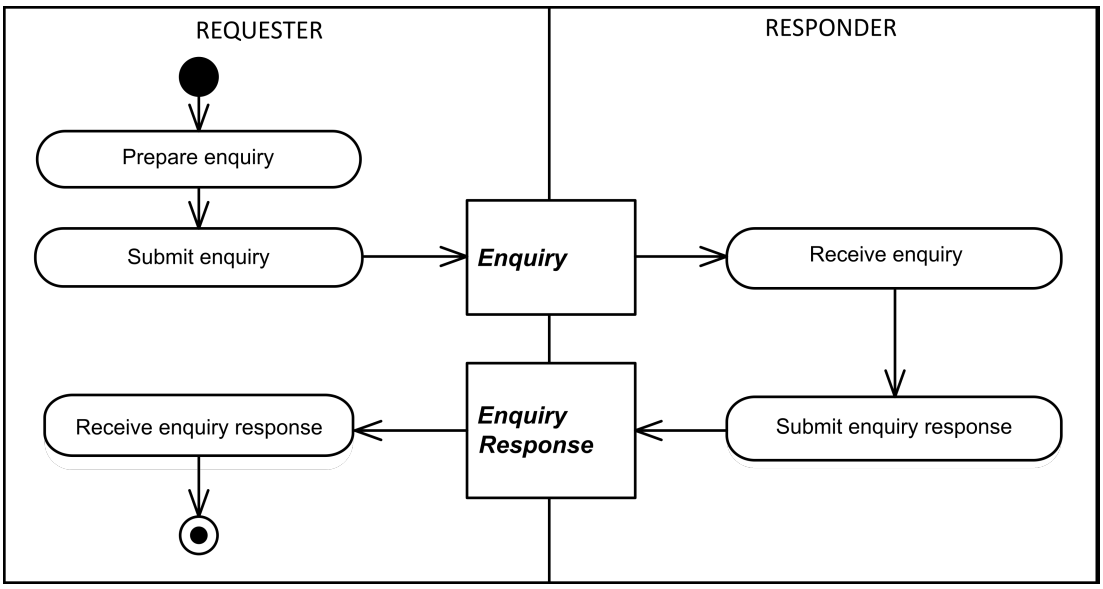

Figure 25 — Enquiry

5.3.3.1.10 Submission of Tenders

A Tenderer submits one or more Tender documents that offer a tender to the Contracting Authority for bid. The Contracting Authority responds with a Tender Receipt to *notify the reception of the tender* for a tendering process. The date and time of the Tender Receipt are significant, because tendering procedures usually have strict deadlines for tender presentation.

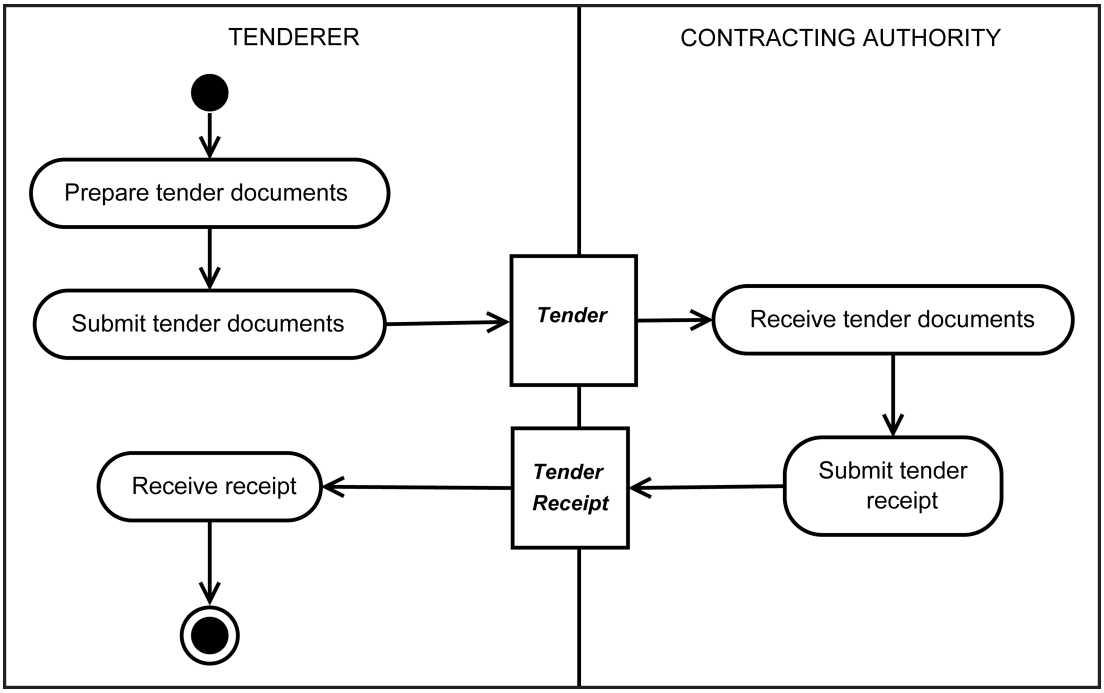

Figure 26 — Submission of Tenders

5.3.3.1.11 Tender Status

An economic operator asks about the details and the status of a tendering procedure. In reply to this enquiry, the contracting authority sends information to the economic operator describing the status of a tendering process.

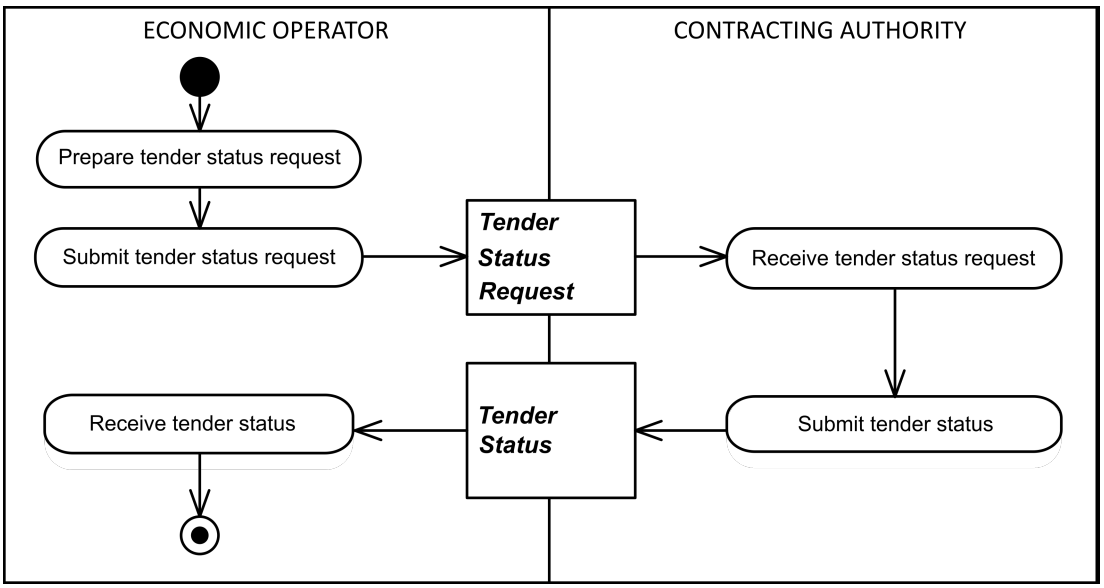

Figure 27 — Tender Status

5.3.3.1.12 Tender Withdrawal

An economic operator requests to withdraw a submitted tender to the contracting authority. Based on that document, the contracting authority will remove the tender from the tendering system.

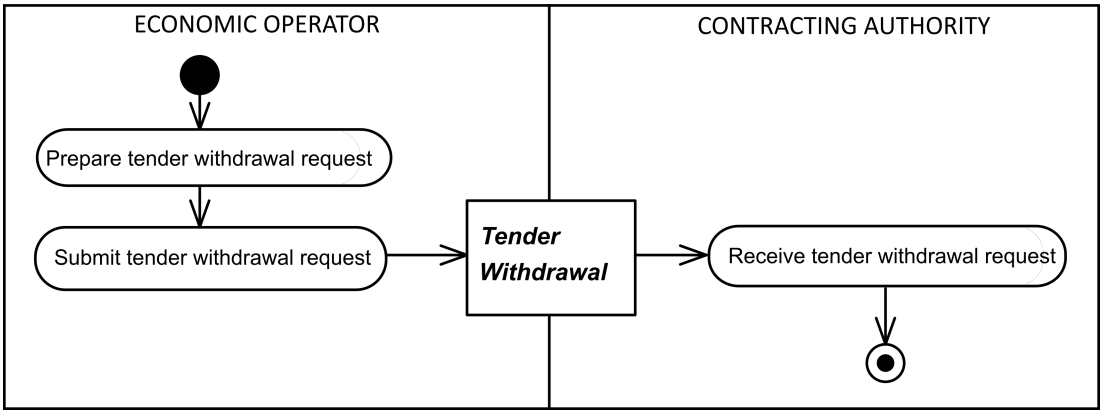

Figure 28 — Tender Withdrawal

5.3.3.1.13 Awarding of Tenders

The awarding of tenders takes place in three phases.

First, the Contracting Authority *notifies each tenderer of its success or failure* in winning the contract, using the Awarded Notification document to communicate the contract award to the winning tenderer or the Unawarded Notification document to communicate that the contract has been awarded to another tenderer.

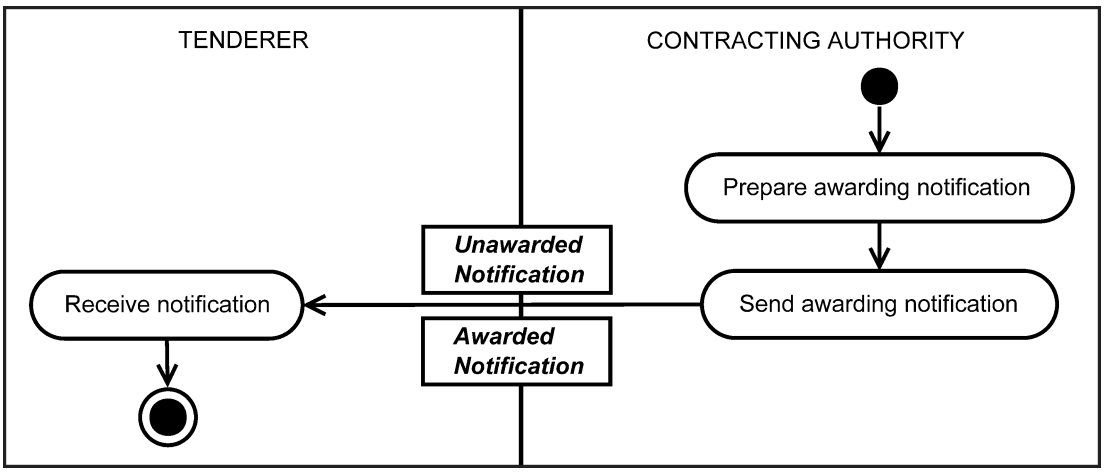

Figure 29 — Award Notification

Second, the Contracting Authority creates a Contract Award Notice to *announce the awarding* of a procurement project.

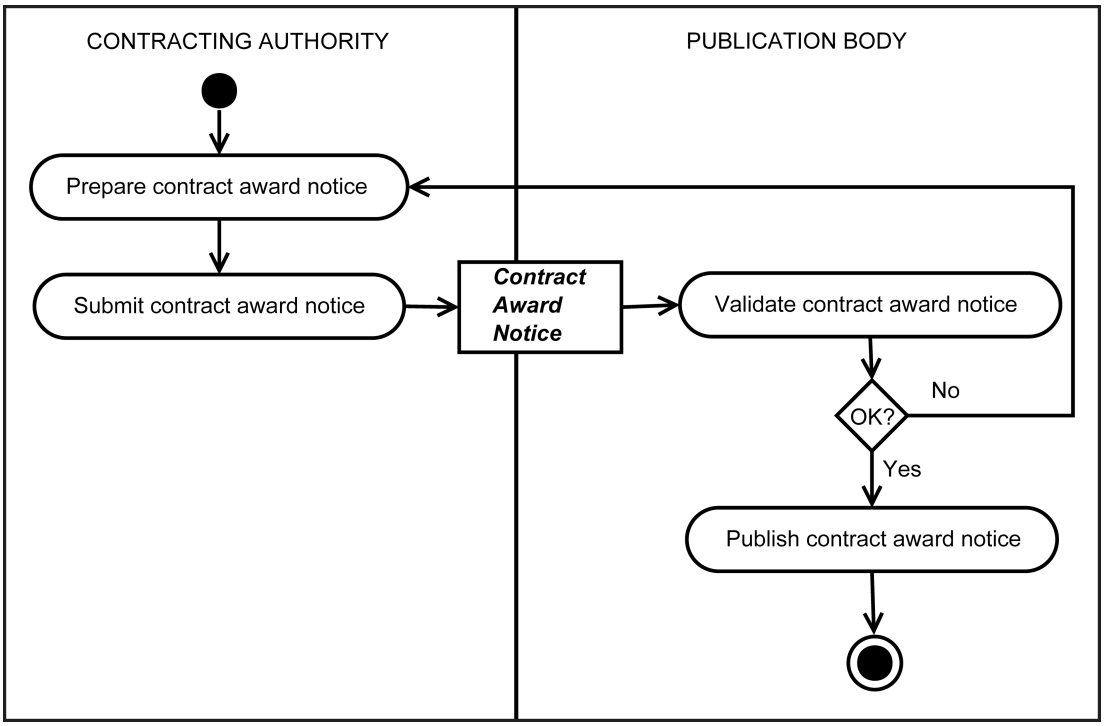

Figure 30 — Award Publication

Finally, the Tenderer sends a Guarantee Certificate to *notify the deposit of a guarantee.*

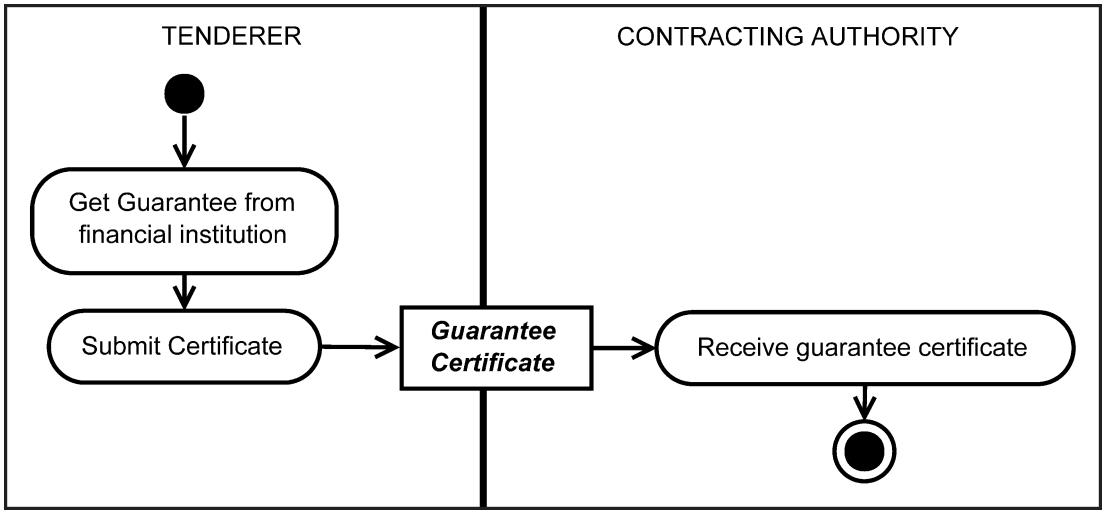

Figure 31 — Guarantee Deposit

5.3.3.1.14 Tender Contract

A process whereby a Contracting Authority sends information to the Economic Operator describing the final contract after a tendering process has been awarded.

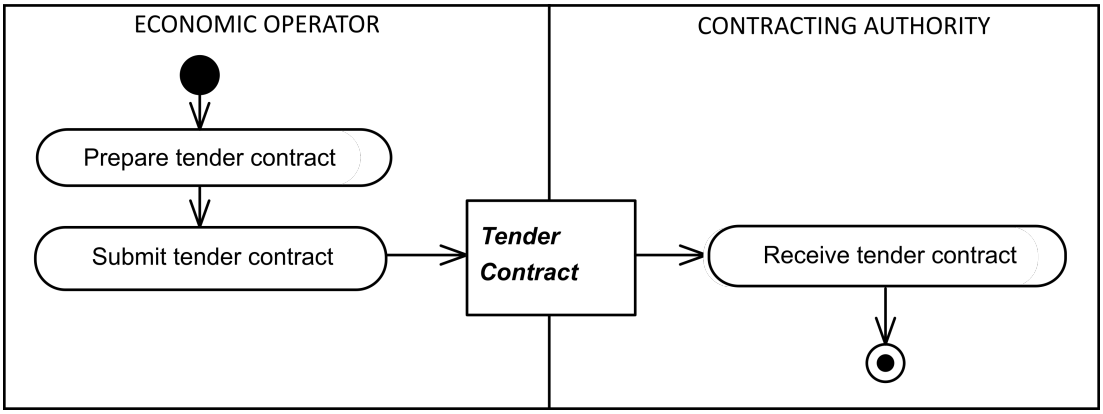

Figure 32 — Tender Contract

5.3.3.2 Catalogue

5.3.3.2.1 Catalogue Introduction

A Catalogue is a document with structured item information that is used for commercial purposes over a period of time. It can be established, updated and deleted with different types of catalogue transactions. Different meanings are given to the concept of catalogue depending on the user's perspective. Thus, it can also be understood as:

— a tender: an electronic document which contains all references of items, services and prices available, proposed by the Catalogue Provider.

— a set of needs: list of needs of products or services that the Customer Party may purchase or contract.

— a requirement: list of selected items and corresponding prices supplied to a Catalogue Provider to be bought.

Document types associated with Catalogue processes are Catalogue Request, Application Response, Catalogue Item Specification Update, Catalogue Pricing Update, and Catalogue Deletion.

5.3.3.2.2 Catalogue Business Rules

— Any conditions specified in the contract shall overrule those stated in the common Catalogue.

— A Catalogue exchange shall be between one Provider and one Receiver Party.

— A classification system may have its own set of properties.

— A classification scheme shall have metadata.

— A Catalogue may have a validity period.

— A Catalogue should include item classifications.

— Classification schemes should include standard and specific properties.

— A Catalogue may refer to the lot (sub-section) of a contract.

— A Catalogue may explicitly specify the framework contract reference.

— A Catalogue may refer to a DPS contract number.

— When a Catalogue item is updated, the item shall be replaced in the Catalogue.

— When a Catalogue item is updated, historical information about replaced or updated items must be available to reconcile with outstanding transactions.

— Prices may be updated independently of other Catalogue information.

— Catalogue distribution may be Provider or Receiver Party initiated.

— If a Receiver initiates a request for a Catalogue, they may request an entire Catalogue or only updates to either pricing or item specification details.

— Whether Receiver Party initiated or not, the decision to issue a new Catalogue or update an existing one shall be at the discretion of the Provider Party.

— If an updated Catalogue is issued, then an action code shall define the status of the items in the Catalogue.

5.3.3.2.3 Catalogue Provision

5.3.3.2.3.1 Catalogue Provision Introduction

Catalogue provision is the case where a Provider sends information regarding items available for purchase to a Receiver. This may be on request or unsolicited. Because they are only potential purchasers, a Receiver may never become a Customer Party.

5.3.3.2.3.2 Create Catalogue

The process of creating a Catalogue is shown in the following diagram. The UBL document types involved are Catalogue, Catalogue Request, and Application Response.

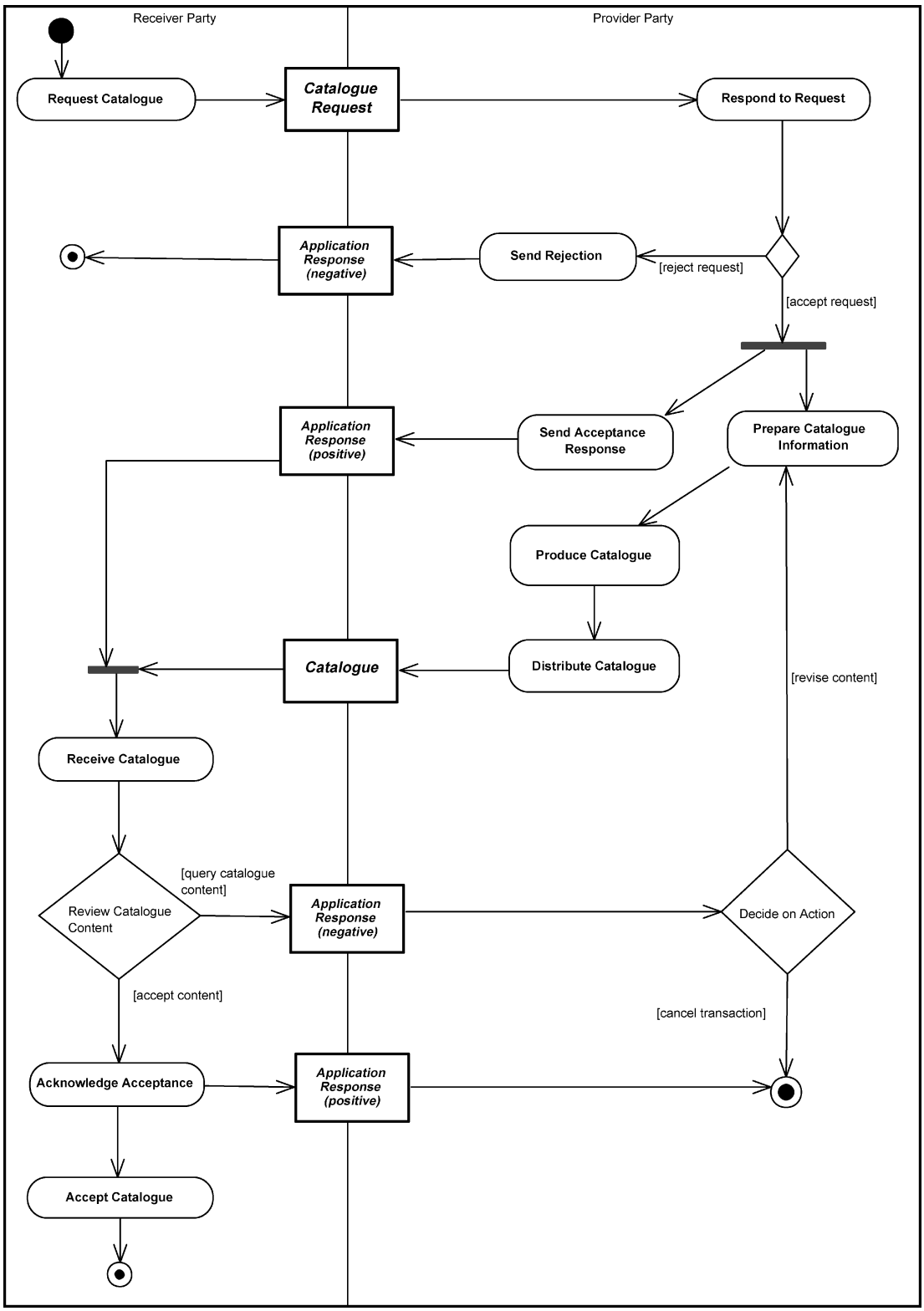

Figure 33 — Create Catalogue Process

5.3.3.2.3.3 Update Catalogue Item Specification

The process of updating a Catalogue Item specification using Catalogue Item Specification Update is shown in the following diagram. The Catalogue Request and Application Response documents also participate.

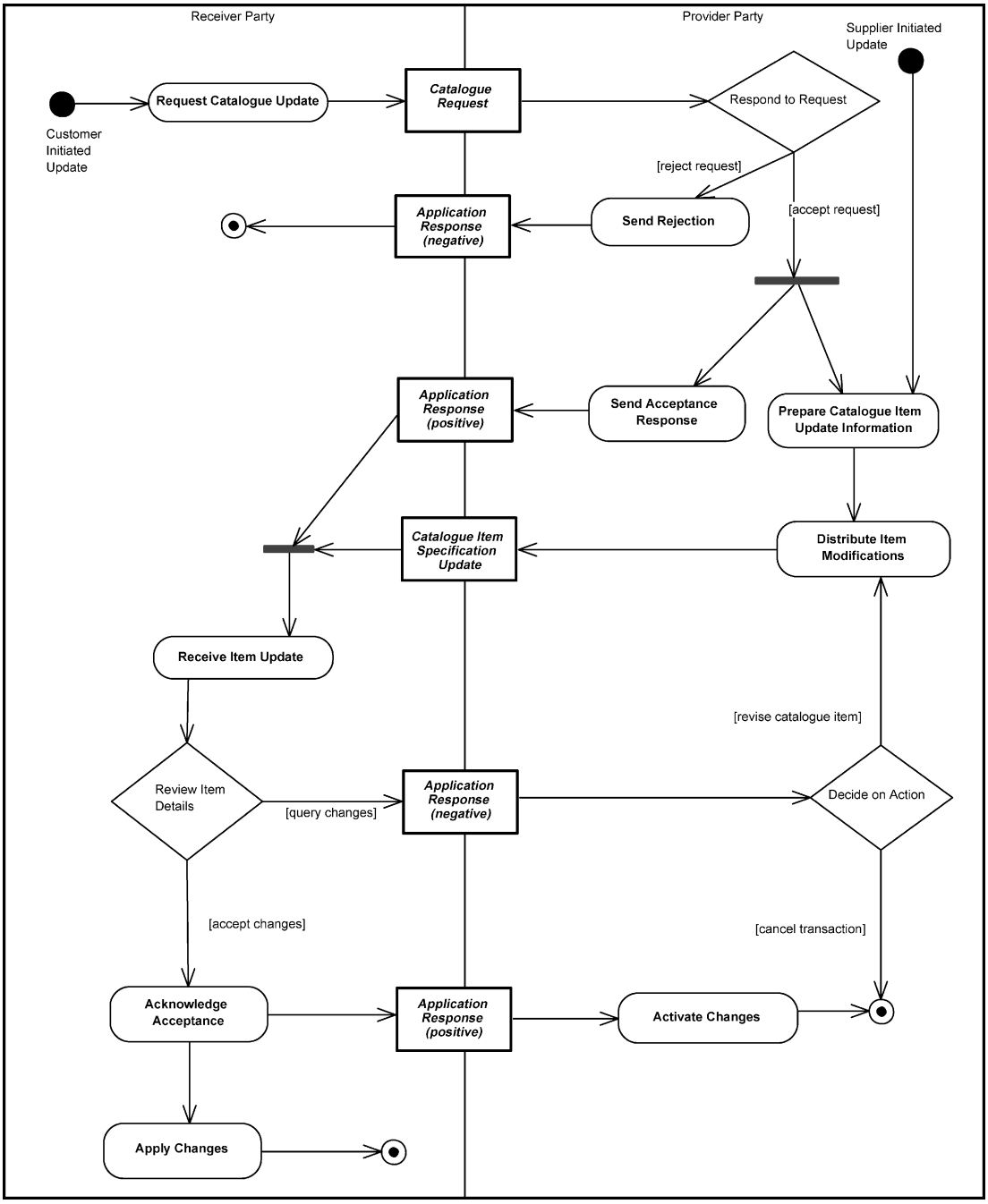

Figure 34 — Update Item Specification Process

5.3.3.2.3.4 Update Catalogue Pricing

The process of updating Catalogue pricing is shown in the following diagram. The UBL document types involved are Catalogue, Catalogue Request, Catalogue Pricing Update, and Application Response.

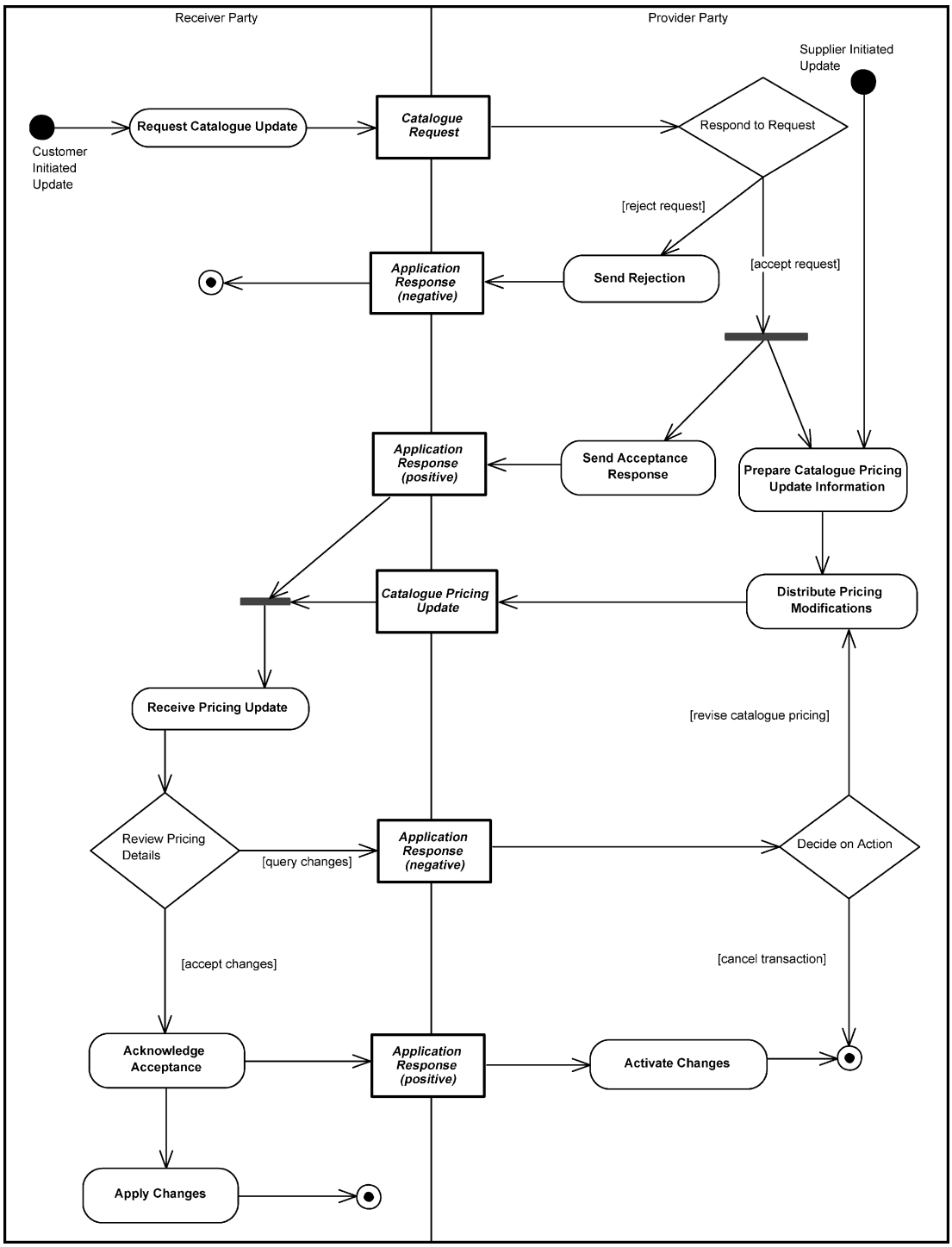

Figure 35 — Update Catalogue Pricing Process

5.3.3.2.3.5 Delete Catalogue

Deletion of a Catalogue using Catalogue Deletion and Application Response is shown in the following diagram.

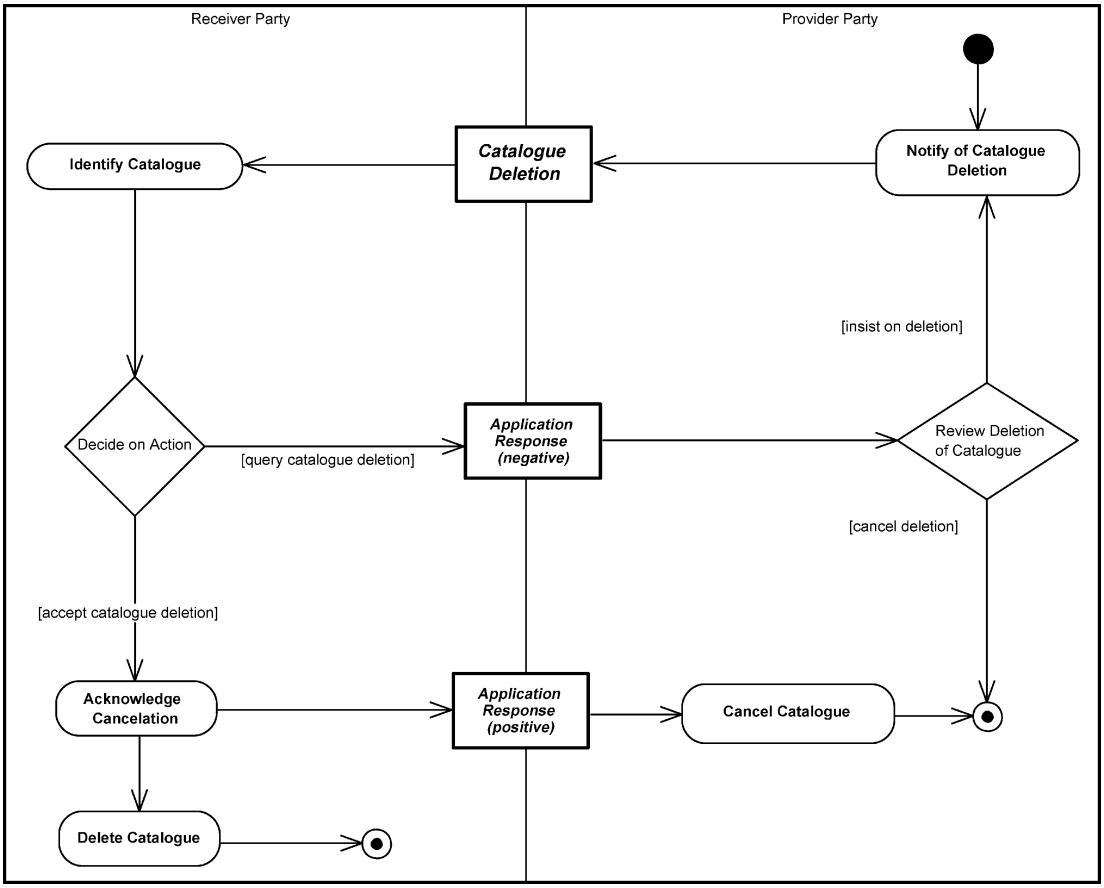

Figure 36 — Delete Catalogue Process

5.3.3.2.3.6 Punchout

Punch-out is a technological innovation whereby an Originator is able to directly access a Seller's catalogue application from within the Seller's own procurement application.

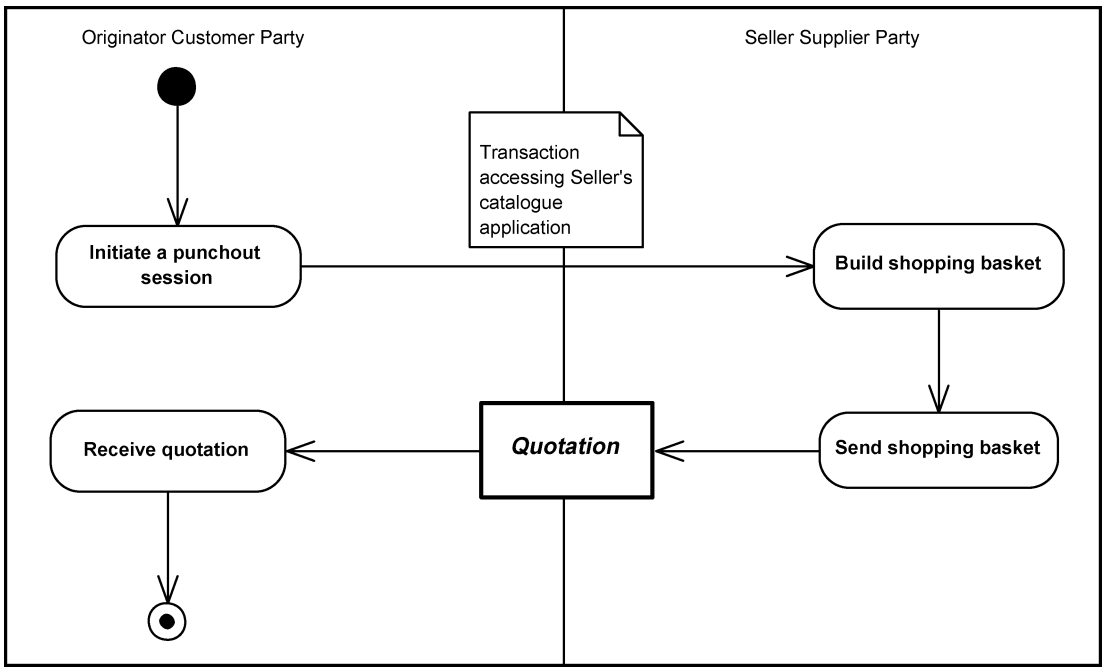

Figure 37 — Punch-out Sourcing Process

The Originators leave ("punch out" from) their system and interact with the Seller's catalogue to locate and order products, while the Seller's procurement application transparently gathers pertinent information.

While conceptually the punch-out request is a form of Request For Quotation (see 5.3.3.3 Quotation), the exchange transaction is tightly coupled to the specific catalogue application and is considered outside the scope of UBL; thus, the only UBL document type involved in this process is Quotation.

5.3.3.3 Quotation

Less formally defined than a tender (see 5.3.3.1 Tendering (pre-award)), a quotation process is the case where the Originator asks for a Quotation via a Request For Quotation, as shown in the following diagram.

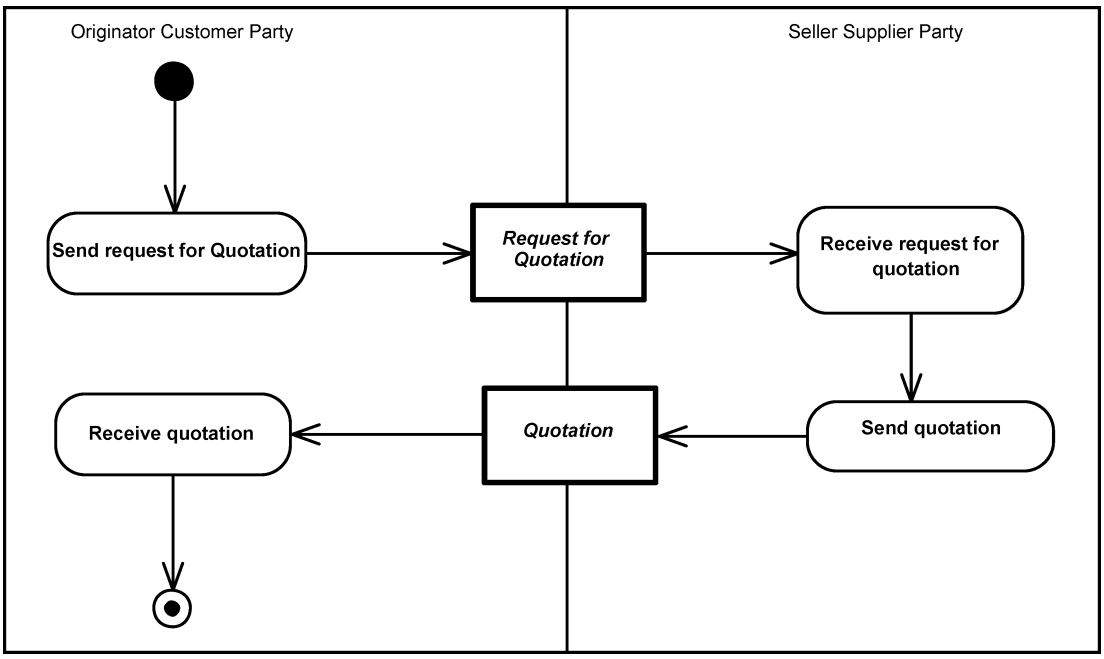

Figure 38 — Quotation Process

5.3.3.4 Ordering (post-award)

5.3.3.4.1 Ordering Introduction

Ordering is the collaboration that creates a contractual obligation between the Seller Supplier Party and the Buyer Customer Party. Document types in these processes are Order, Order Response, Order Response Simple, Order Change, and Order Cancellation.

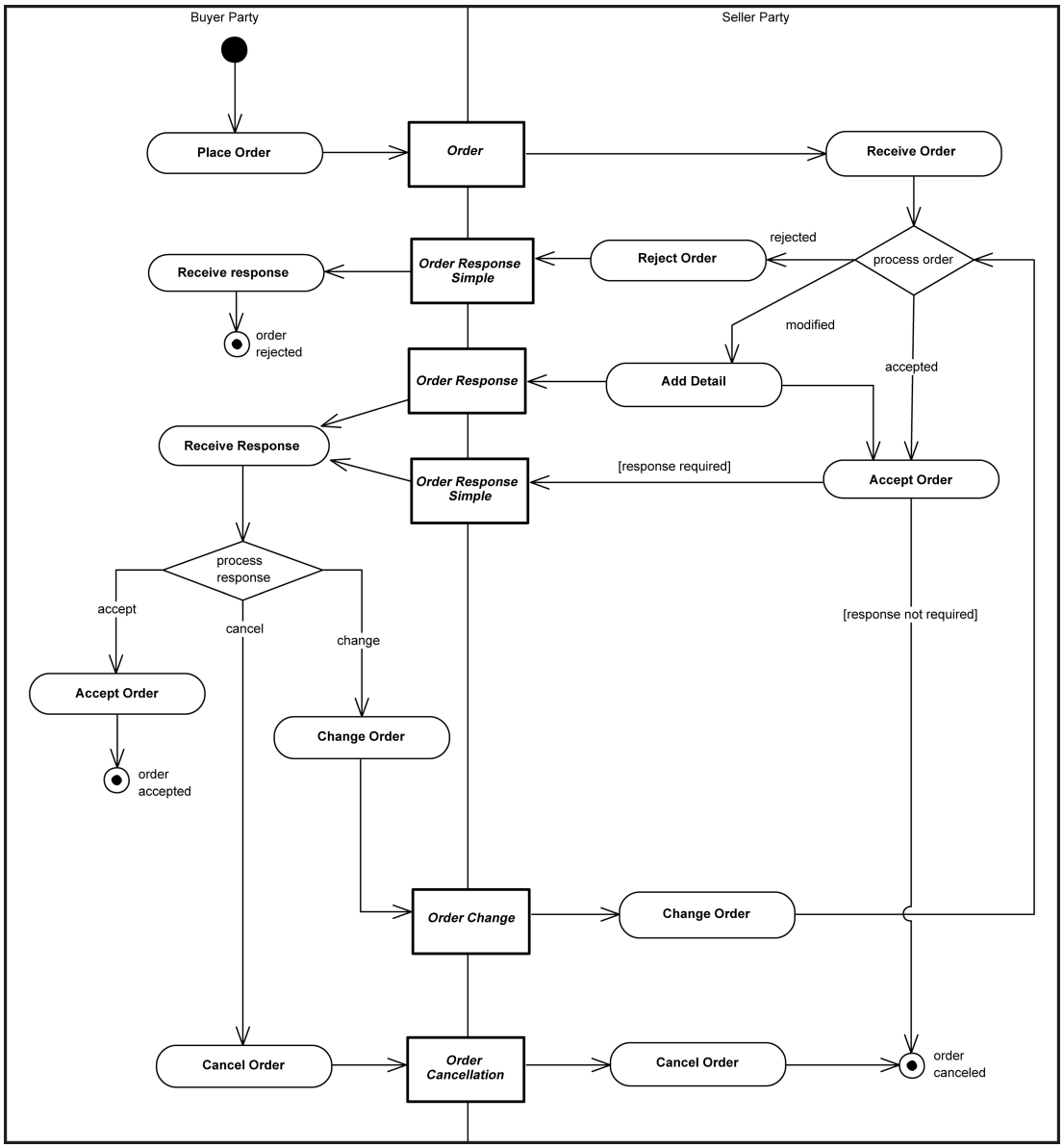

Figure 39 — Ordering Process

5.3.3.4.2 Ordering Business Rules

— The Order may specify allowance and charge instructions (e.g., freight, documentation, etc.) that identify the type of charge and who pays which charges. The Order may be placed "on account" against a trading credit account held by the Seller, or against a credit/debit card account, or against a direct debit agreement. The Order allows for an overall currency defining a default for all pricing and also a specific currency to be used for Invoicing. Within an Order, additional currencies may be specified both for individual item pricing and for any allowances or charges.

— Trade discount may be specified at the Order level. The Buyer may not know the trade discount, in which case it is not specified. This makes a detailed response from the Seller necessary; see 5.3.3.4.4 Order Response.

— The Order provides for multiple Order Lines.

— The Order may specify delivery terms, while the Order Line may provide instructions for delivery.

— The Buyer may indicate potential acceptable alternatives.

5.3.3.4.3 Order Response Simple

The Order Response Simple is the means by which the Seller confirms receipt of the Order from the Buyer, indicating either commitment to fulfil without change or that the Order has been rejected.

5.3.3.4.4 Order Response

Proposed changes to an Order by the Seller are accomplished through the full Order Response document.

The Order Response proposes to replace the original Order. It reflects the entire new state of an order transaction.

It also is the means by which the Seller confirms or supplies Order-related details to the Buyer that were not available to, or specified by, the Buyer at the time of ordering. These may include:

— Delivery date, offered by the Seller if not specifically requested by the Buyer

— Prices

— Discounts

— Charges

— Item Classification codes

The Seller may advise on replacements, substitutes, or other necessary changes using the Order Response.

5.3.3.4.5 Order Change

The Buyer may change an established Order in two ways, subject to the legal contract or trading partner agreement: first, by sending an Order Change, or second, by sending an Order Cancellation (see 5.3.3.4.6 Order Cancellation) followed by a new, complete replacement Order.

An Order Change reflects the entire current state of an order transaction.

Buyers may initiate a change to a previously accepted order for various reasons, such as changing ordered items, quantity, delivery date, ship-to address, etc. Suppliers may accept or reject the Order Change using either Order Response or Order Response Simple.

5.3.3.4.6 Order Cancellation

At any point in the process, a Buyer may cancel an established order transaction using the Order Cancellation document. Legal contracts, trading partner agreements, and business rules will determine the point at which an Order Cancellation will be ignored (e.g., at the point of manufacture or the initiation of the delivery process). Given the agreements and rules, an Order Cancellation may or may not be an automated business transaction. The terms and conditions of contract formation for business commitments will dictate which, if any, of these restrictions or guidelines will apply.

5.3.3.5 Vendor Managed Inventory

5.3.3.5.1 Vendor Managed Inventory Introduction

Vendor Managed Inventory (VMI) is a family of business processes in which the Retailer Customer Party for an item provides certain information to the Seller Supplier Party, and the Seller Supplier Party takes full responsibility for maintaining an agreed-upon inventory of the item, usually at the Retailer Customer Party's point of sale. A third party logistics provider can also be involved to make sure that the Retailer Customer Party has the required level of inventory by adjusting the demand and supply gaps.

UBL supports three common models of VMI:

— Basic VMI

— Cyclic Replenishment Program (CRP)

— Replenishment on Customer Demand

These processes are described in more detail below. It should be noted that the particular semantics used here come from a large-scale UBL application developed for the Italian textile and clothing industry by ENEA, the Italian National Agency for New Technologies, Energy, and Sustainable Economic Development (see [19]). These models are applicable to the implementation of vendor-managed relationships in a broad range of retail sectors, but for the sake of simplicity, and in keeping with the model application, the two principal parties in the VMI relationship (the Seller Supplier Party and the Retailer Customer Party) are referred to as "producer" and "retailer" in the descriptions that follow; more generically, they are vendor and customer.

5.3.3.5.2 Basic Vendor Managed Inventory

5.3.3.5.2.1 Basic Vendor Managed Inventory Introduction

In the classic VMI scenario, a shop-within-a-shop area or an entire store is managed completely by the producer. The logistic concept of VMI can be combined with consignment/concession as well as with charge-on-delivery as the financial model. Mostly it is combined with consignment.

5.3.3.5.2.2 Initial Stocking of the Area by Producer

At the beginning of the cooperation, the area is stocked by the producer. The retailer receives item and delivery information and reports back the goods actually received. UBL document types used here are Catalogue, Despatch Advice, and Receipt Advice.

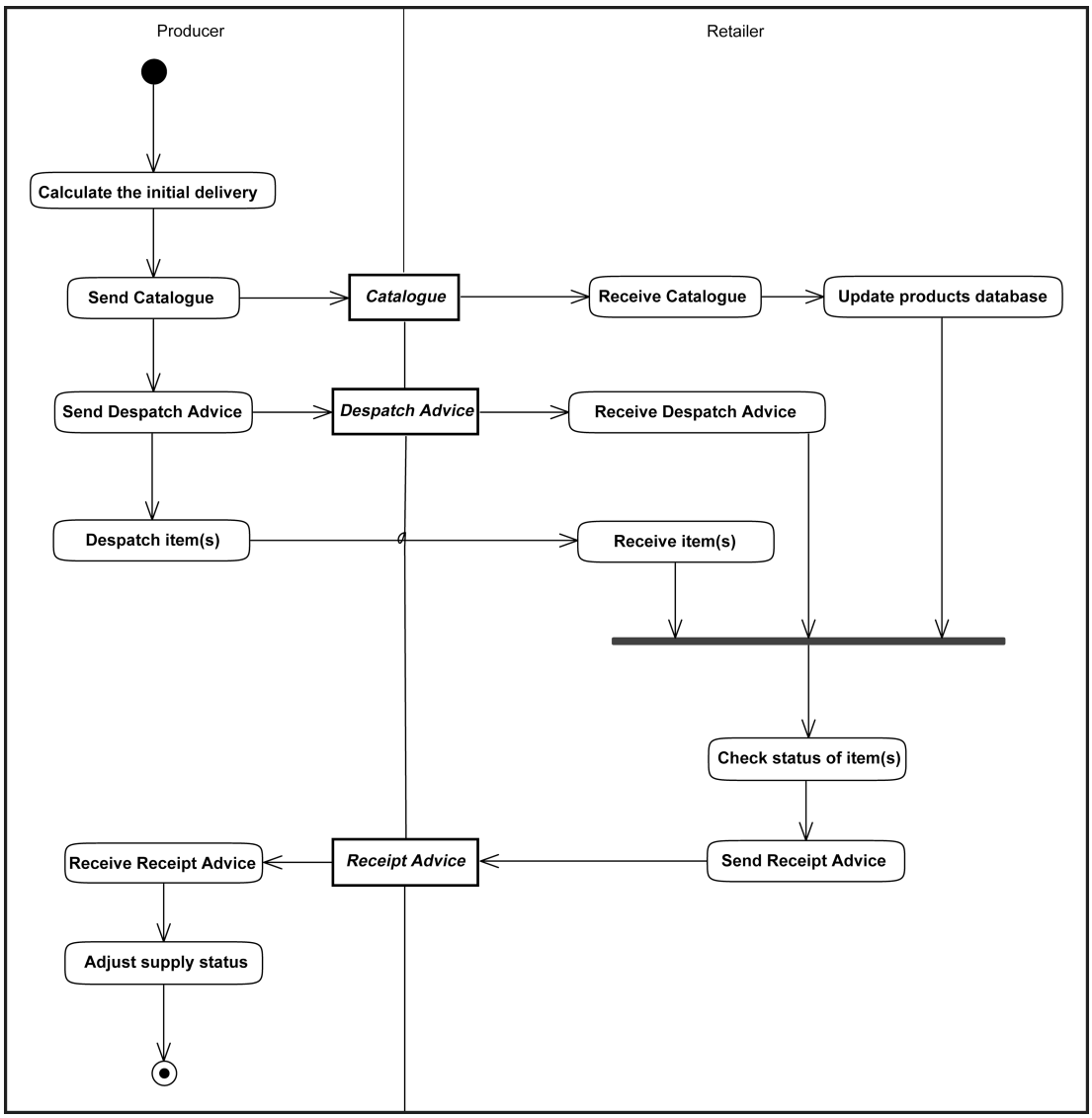

Figure 40 — Initial Stocking of the Area by Producer

5.3.3.5.2.3 Report of Sales and Inventory Movement

The sales and inventory movement information is transferred from the retailer to the producer using Product Activity.

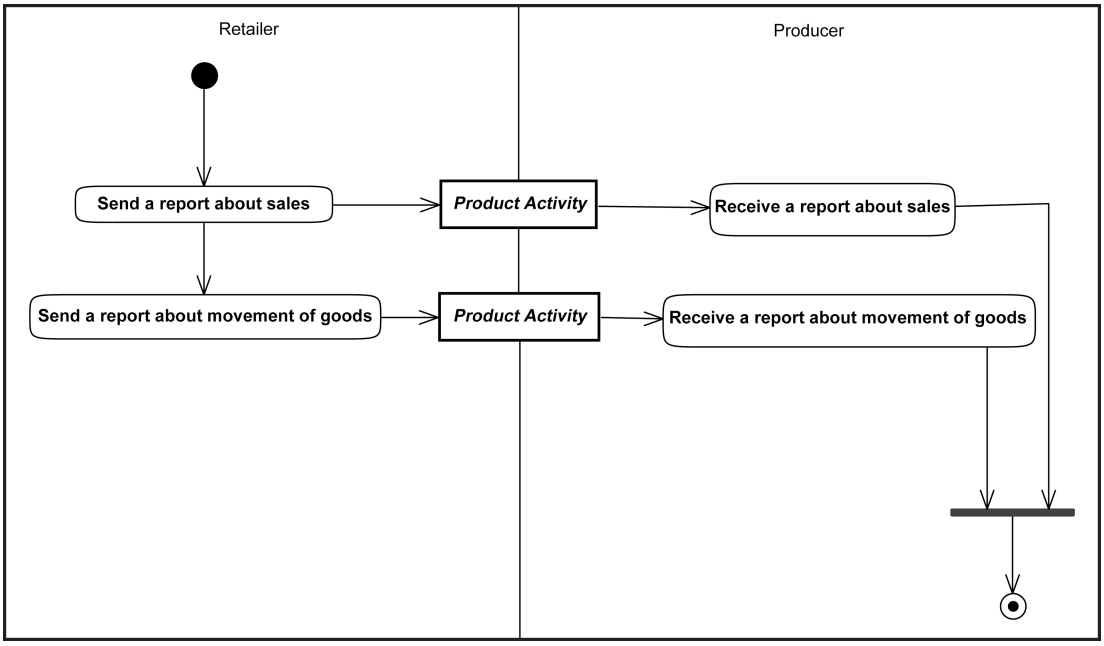

Figure 41 — Report of Sales and Inventory Movement

5.3.3.5.2.4 Permanent Replenishment

Based on sales and inventory movement, the producer periodically makes a new delivery of goods accompanied by a Despatch Advice. If the delivery contains an item not previously stocked, an updated Catalogue is also sent so that the retailer can add the item to its product database. Upon delivery of the goods, the retailer reports back the items received using a Receipt Advice.

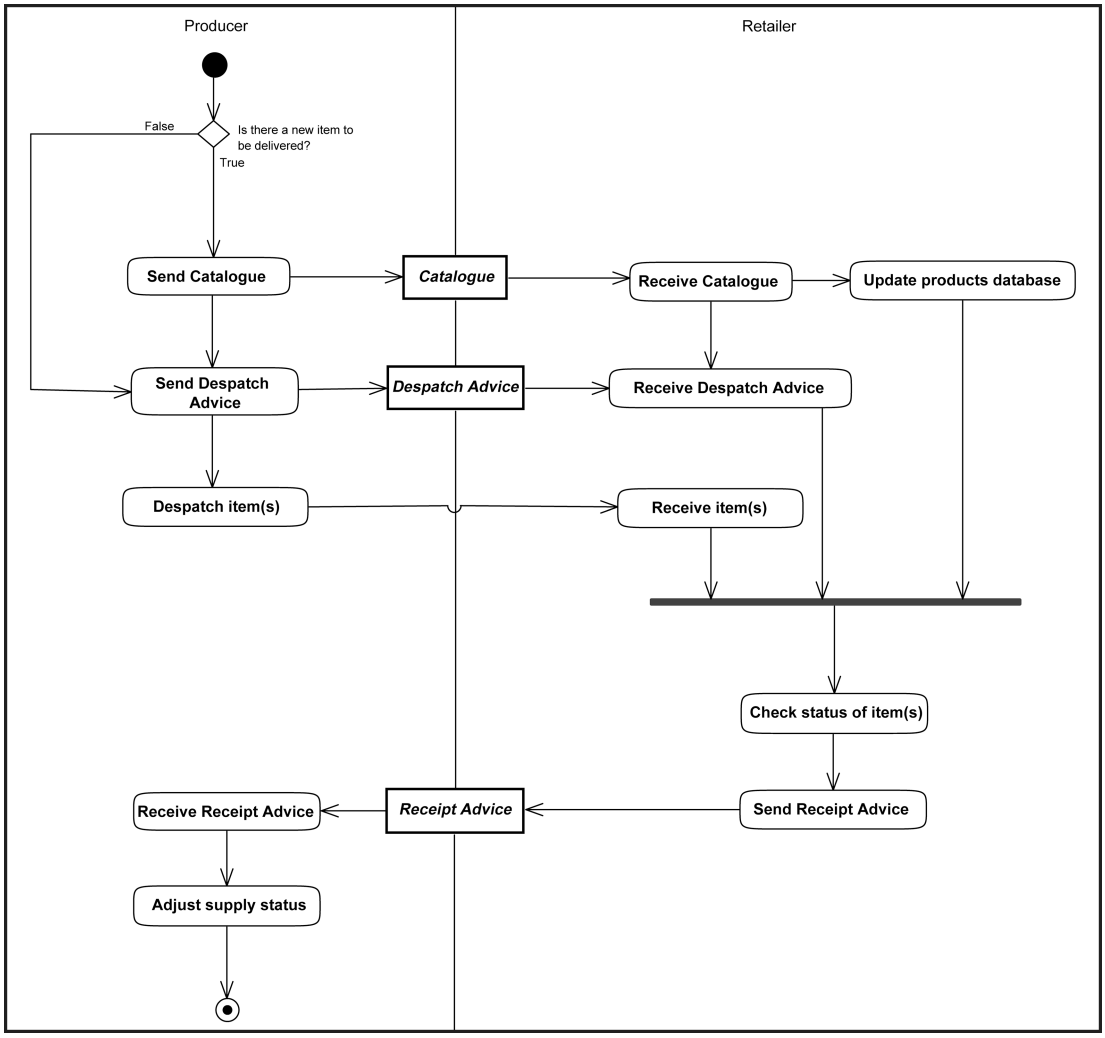

Figure 42 — Permanent Replenishment

5.3.3.5.2.5 Invoicing for Vendor Managed Inventory

A UBL Invoice is sent either on a delivery or a sales basis. In a charge-on-delivery model, the data for the invoice is prepared from the delivery, and in a consignment/concession model from the sales reports.

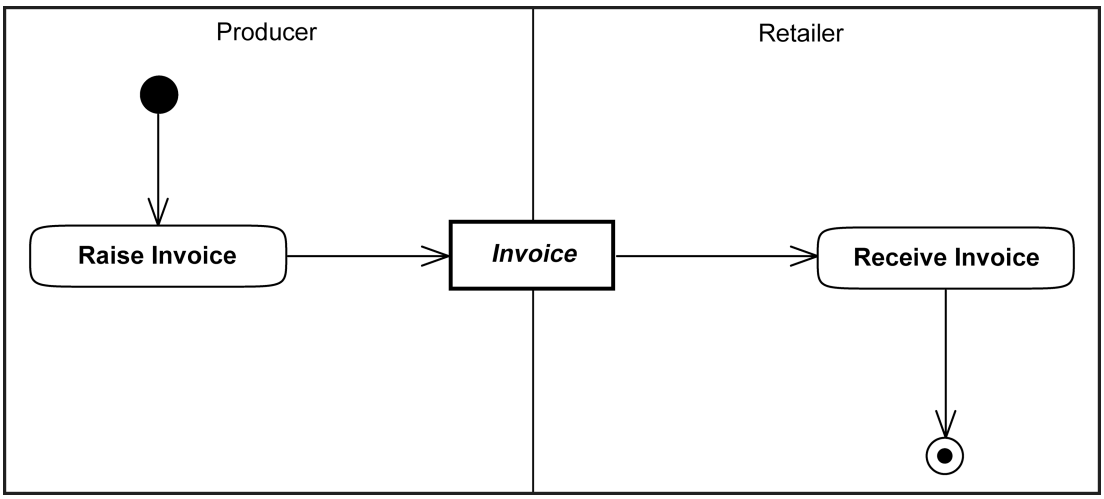

Figure 43 — Invoicing for Vendor Managed Inventory

5.3.3.5.2.6 Returns Initiated by the Producer

If sales do not meet expectations, items are reallocated by the producer. Because the producer cannot request a retailer to send the products to a competitor, the producer requests a return and handles the goods afterwards by itself. Document types used here are Instruction For Returns, Despatch Advice, and Receipt Advice.

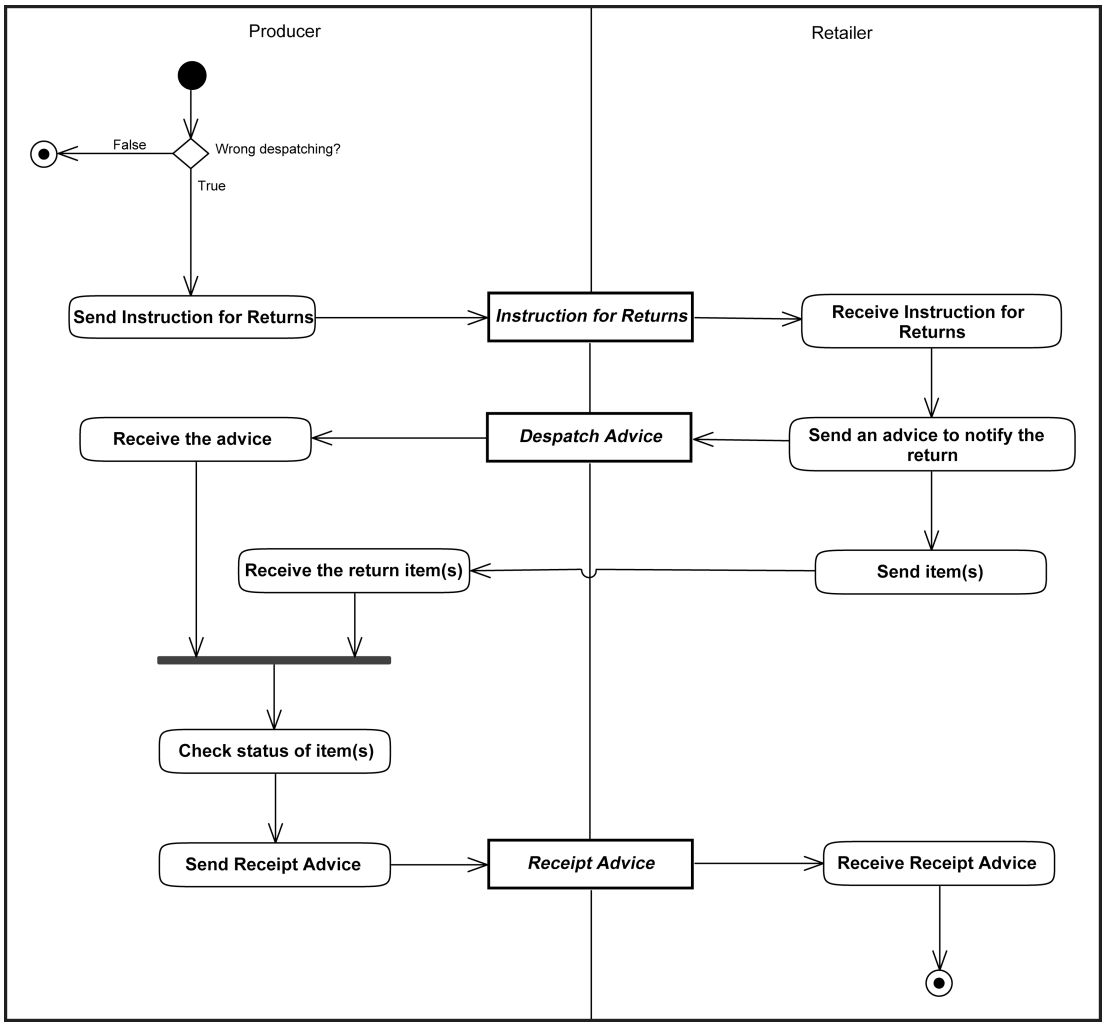

Figure 44 — Returns Initiated by the Producer

5.3.3.5.2.7 Price Adjustments

In the event of a price change, an updated price list (in the form of a new Catalogue containing the change) is sent from producer to retailer.

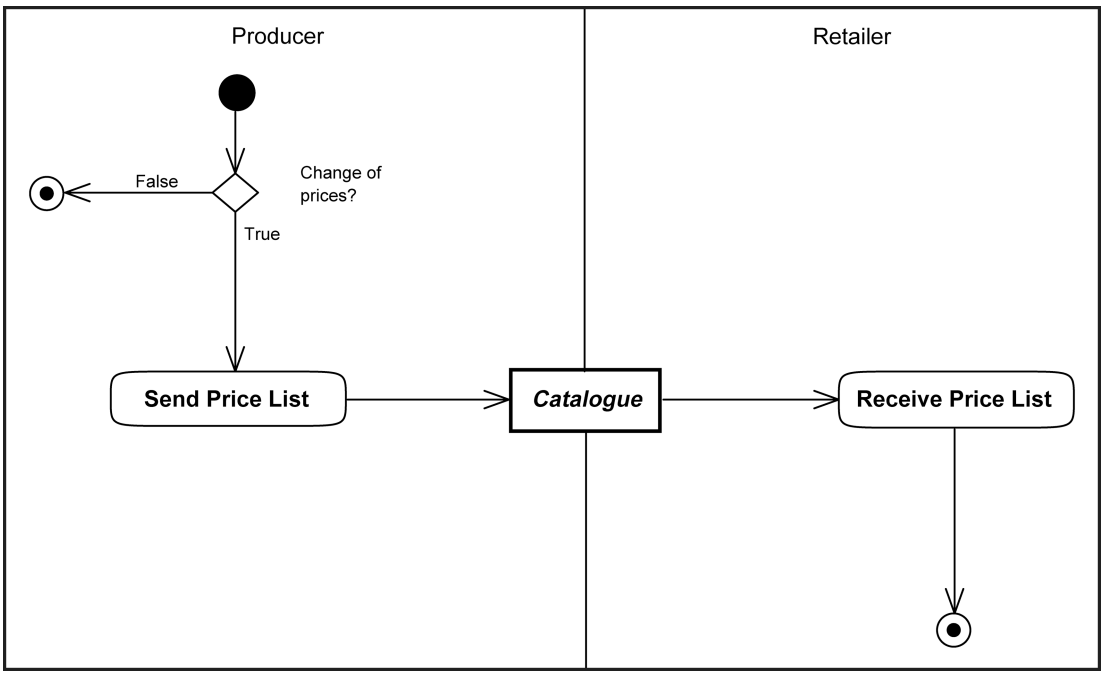

Figure 45 — Price Adjustments

5.3.3.5.3 Cyclic Replenishment Program (CRP)

5.3.3.5.3.1 Cyclic Replenishment Program (CRP) Introduction

A variant of VMI is the Cyclic Replenishment Program (CRP). In this process, the producer establishes a catalogue of NOS (Never Out of Stock) or seasonal NOS items, and the retailer chooses items for cyclic (weekly) replenishment. The logistic scenario can be combined with the charge-on-delivery as well as with a consignment/concession model. At the end of every sales period, a report of sales and inventory movement at all retail locations is sent to the producer.

CRP differs from the third VMI variant, Replenishment on Customer Demand (below), in that the producer cannot change the terms of the order.

5.3.3.5.3.2 Transfer of Base Item Catalogue

The producer publishes the Catalogue of its NOS and seasonal NOS items to the retailer.

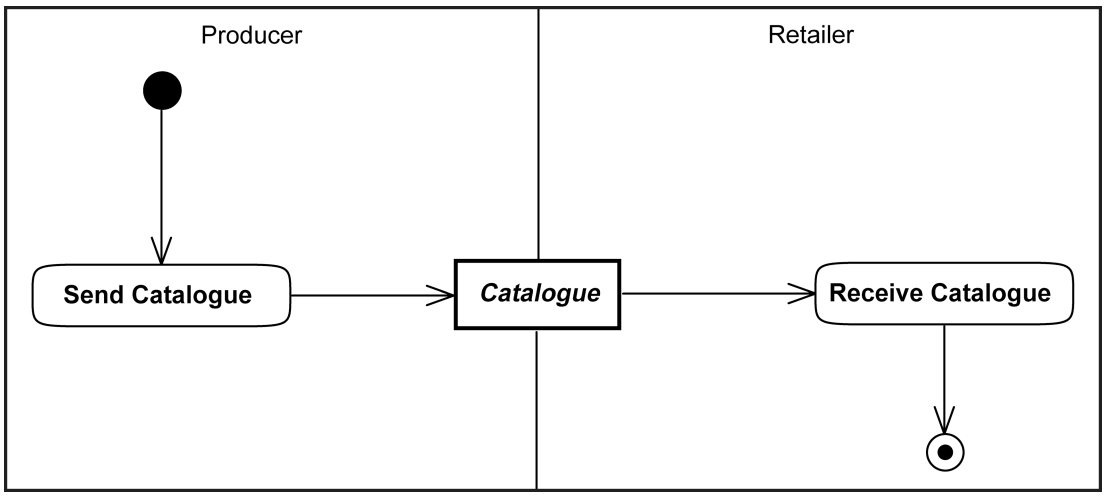

Figure 46 — Transfer of Base Item Catalogue

5.3.3.5.3.3 Initial Stocking of the Area by Retailer

At the beginning of the cooperative relationship—or the beginning of a season, if seasonal NOS products are the focus—the retailer orders its base stock, and the products are delivered. Order, Despatch Advice, and Receipt Advice are used in this process.

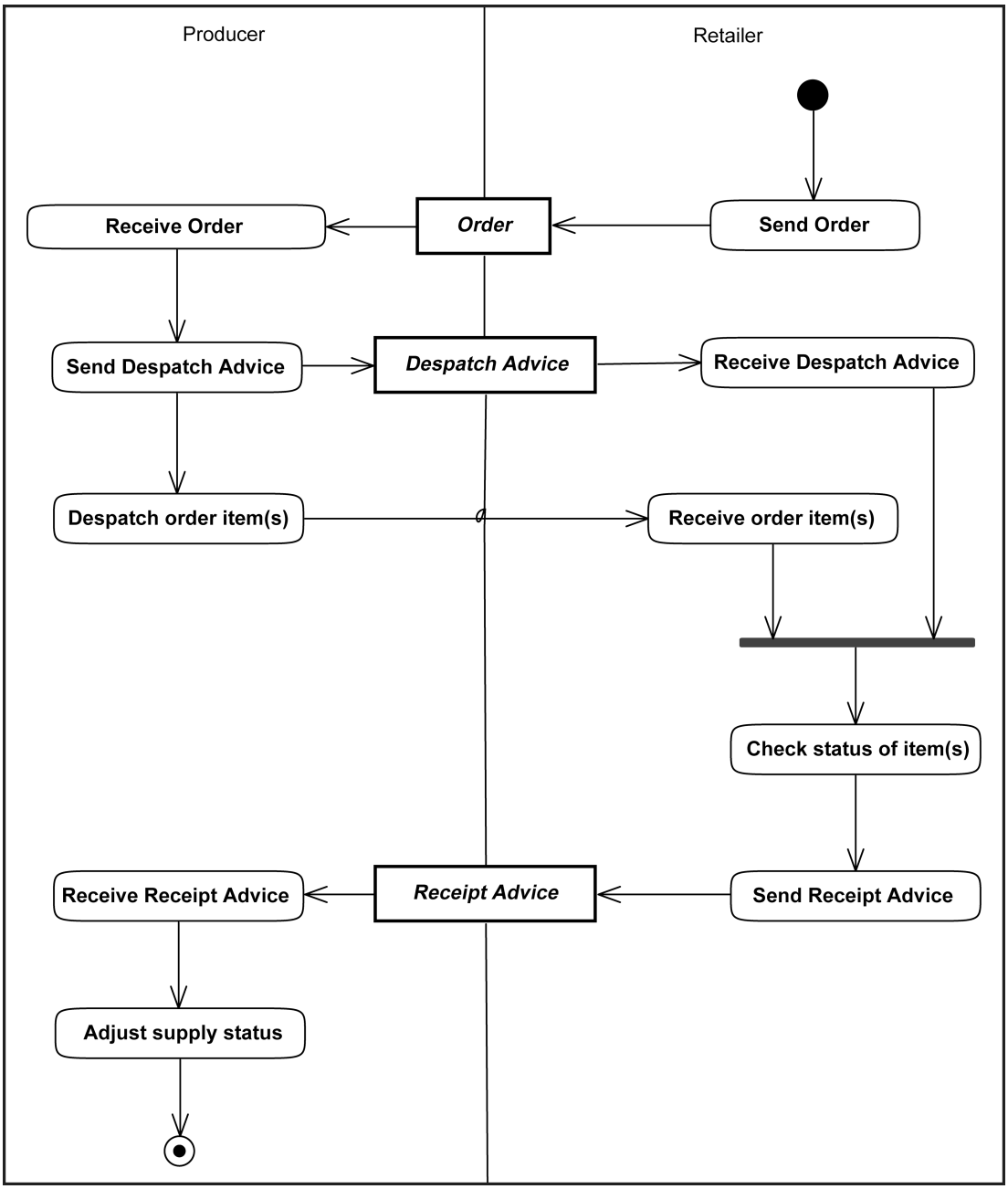

Figure 47 — Initial Stocking of the Area by Retailer

5.3.3.5.3.4 Periodic (Weekly) Replenishment

Each period (every week), the retailer's system calculates the quantities needed for replenishment of the product area. From the result, an order is sent, and the producer responds with a direct delivery within 48 hours.

The replenishment process uses the same documents in the same order as the Initial Stocking process, so the duplicate diagram is omitted here; see Figure 47. It must be remembered, however, that the two processes are taking place at different points in time, so their pre and post conditions will be different.

5.3.3.5.3.5 Report of Sales and Inventory Movements

At the end of each sales day, a report of all sales and inventory movement at all retail locations is sent from the retailer to the producer using Product Activity.

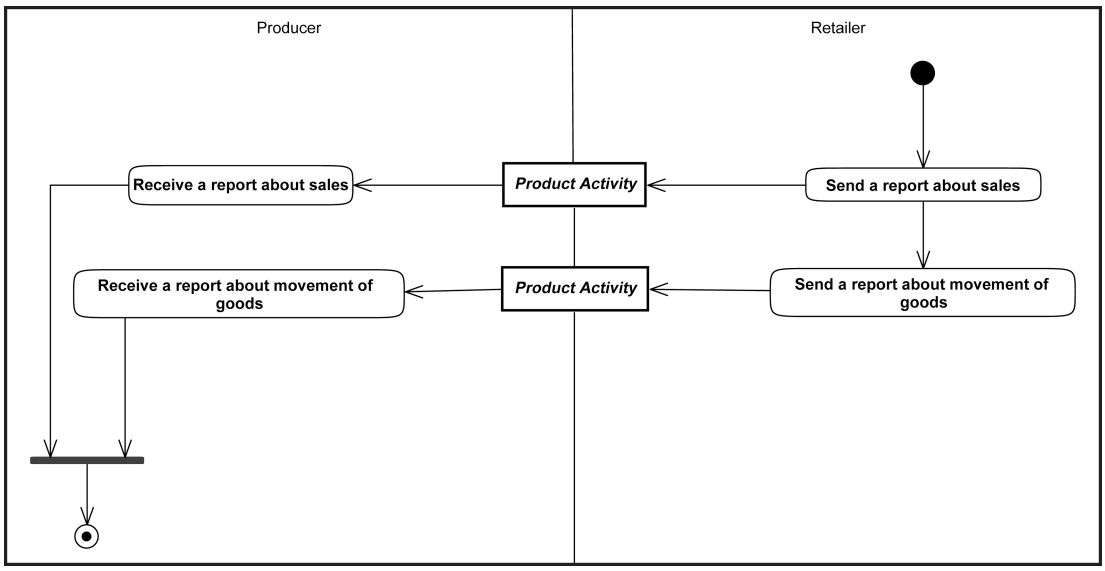

Figure 48 — Report of Sales and Inventory Movements

5.3.3.5.3.6 Cyclic Replenishment Program Invoicing

A UBL Invoice is sent either on a delivery or a sales basis.

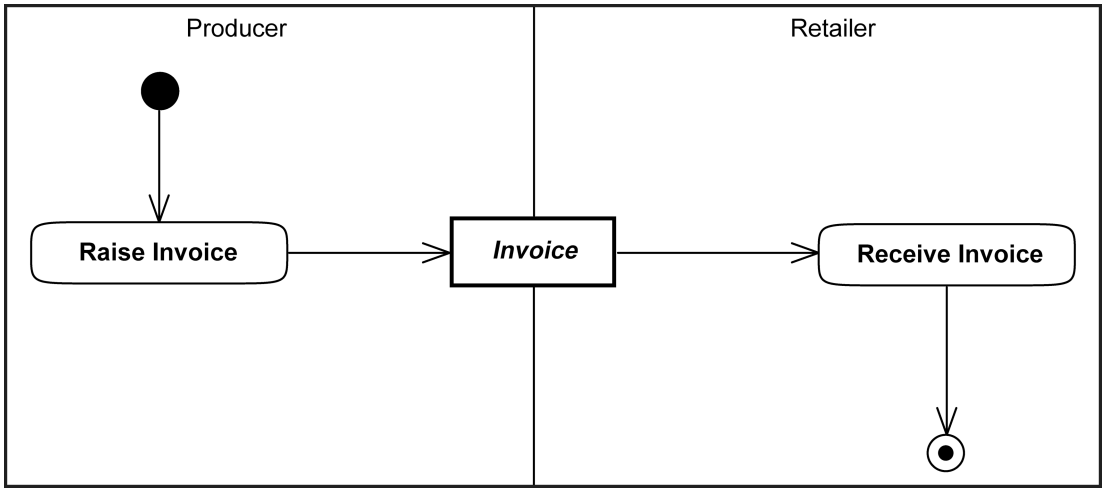

Figure 49 — Invoicing for Cyclic Replenishment Program

5.3.3.5.3.7 Synchronizing of Stock Information

Information about the actual stock is synchronised periodically (for example, every one to three months) using Inventory Report. This is combined at least once a year with a physical inventory.

The retailer sends an inventory report containing the information about the quantities currently in stock.

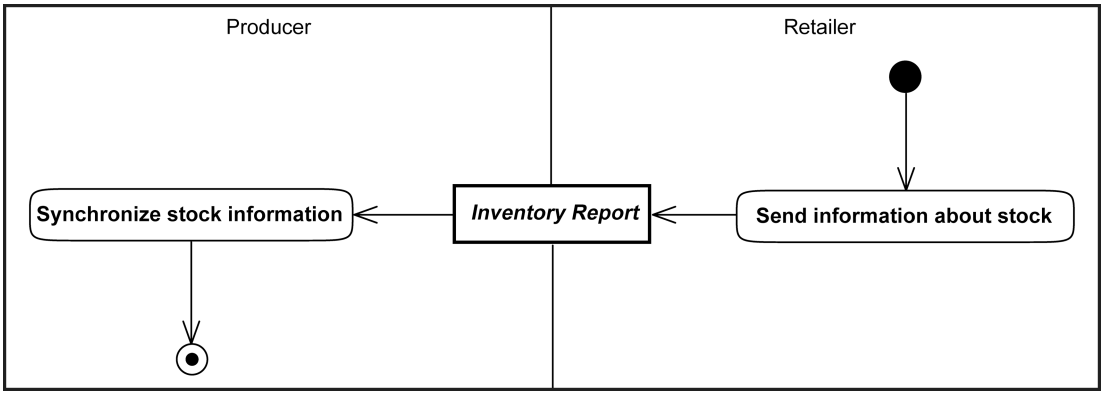

Figure 50 — Synchronizing Stock Information

5.3.3.5.3.8 Changes to the Item Catalogue

In the event of a change, either inside an item belonging to the CRP Catalogue or the relationship of an item to the CRP Catalogue, information about the change is sent to the retailer by sending an updated Catalogue document. Item change is indicated by an optional Action Code field in each changed Catalogue Line.

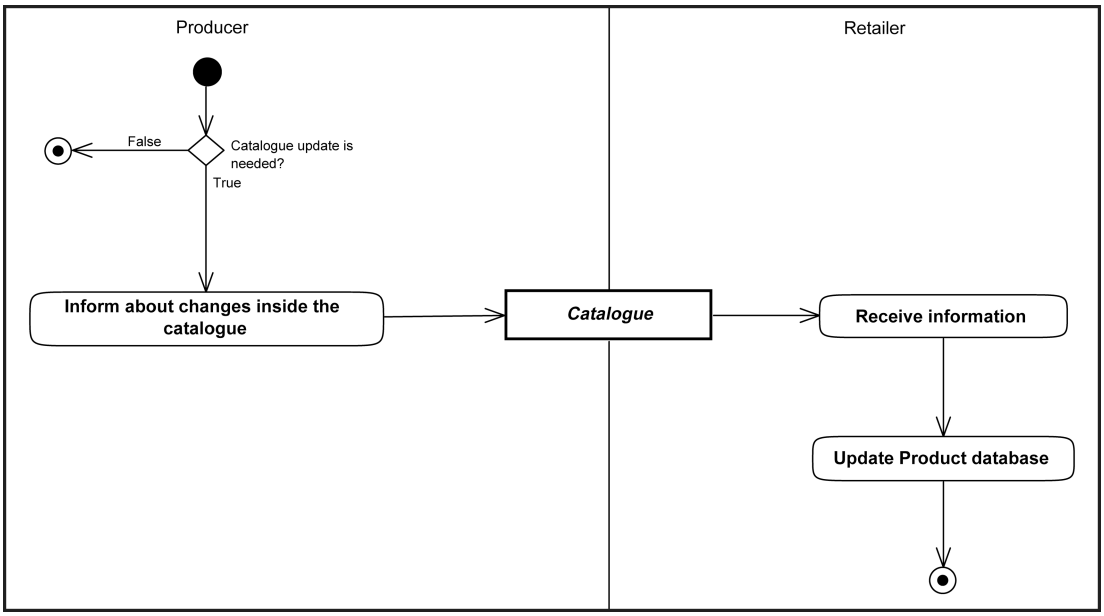

Figure 51 — Changes to the Item Catalogue

5.3.3.5.4 Replenishment On Customer Demand

5.3.3.5.4.1 Replenishment On Customer Demand Introduction

Another variant of VMI is Replenishment On Customer Demand. In this process, the producer selects a subset of its products for a specific retailer and sends out the related article catalogue. Then the producer periodically sends information about the availability of items so that the retailer can form the best ordering plan. The replenishment periodically happens on retailer (customer) demand, and unlike the case with CRP (above), the producer is allowed to propose changes to the orders. Also, because of the requirement to update item availability information, an additional document type (Stock Availability Report) is added to the process.

The processes of sales and inventory reporting, invoicing, stock synchronization, and changing the catalogue are identical to the same processes in CRP. As with CRP, a report of sales and inventory movement at all retail locations is sent to the producer at the end of every sales period. Invoicing and logistics are normally charge-on-delivery but can also be based on a consignment/concession model.

5.3.3.5.4.2 Transfer of Base Article Catalogue

The producer publishes a Catalogue of its products to the retailer. The catalogue can include basic articles, never-out-of-stock (NOS) articles, seasonal articles, short-season-collection articles, or seasonal NOS articles.

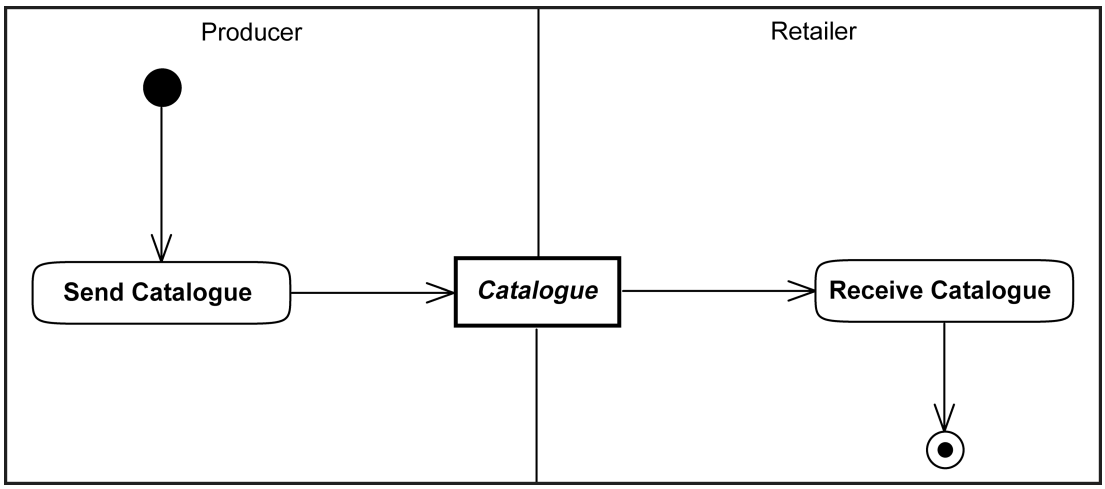

Figure 52 — Transfer of Base Article Catalogue

5.3.3.5.4.3 Periodic Transfer of Article Availability Information

The producer sends out information about availability of goods (quantities on hand, quantities incoming, articles out of stock) using a Stock Availability Report.

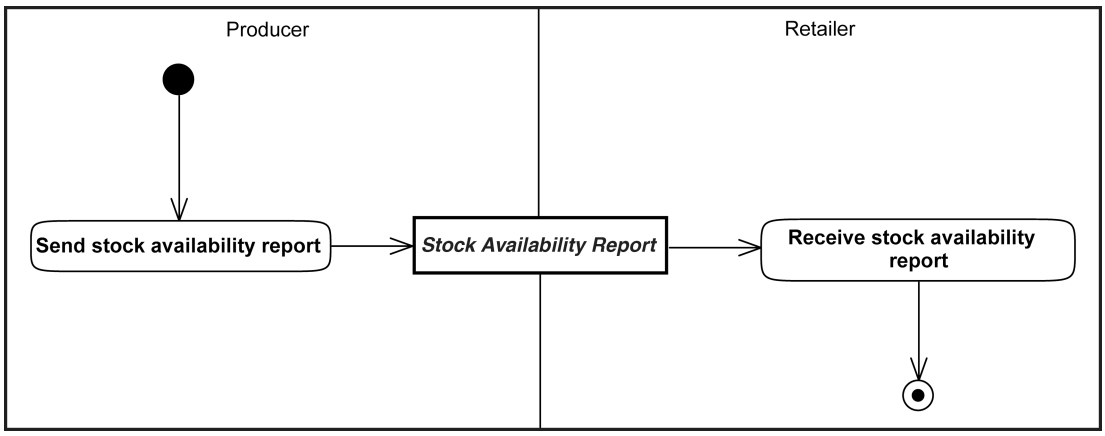

Figure 53 — Periodic Transfer of Article Availability Information

5.3.3.5.4.4 Initial Stocking of the Area by Producer and Retailer

At the beginning of the business cooperation—or perhaps at the beginning of a season, if seasonal NOS (never out of stock) products are the focus—the retailer orders its base stock and the products are delivered. Note that the producer is allowed to propose changes to the order (compare this figure with Figure 47). Document types used in this process include Order, Order Change, Despatch Advice, and Receipt Advice.

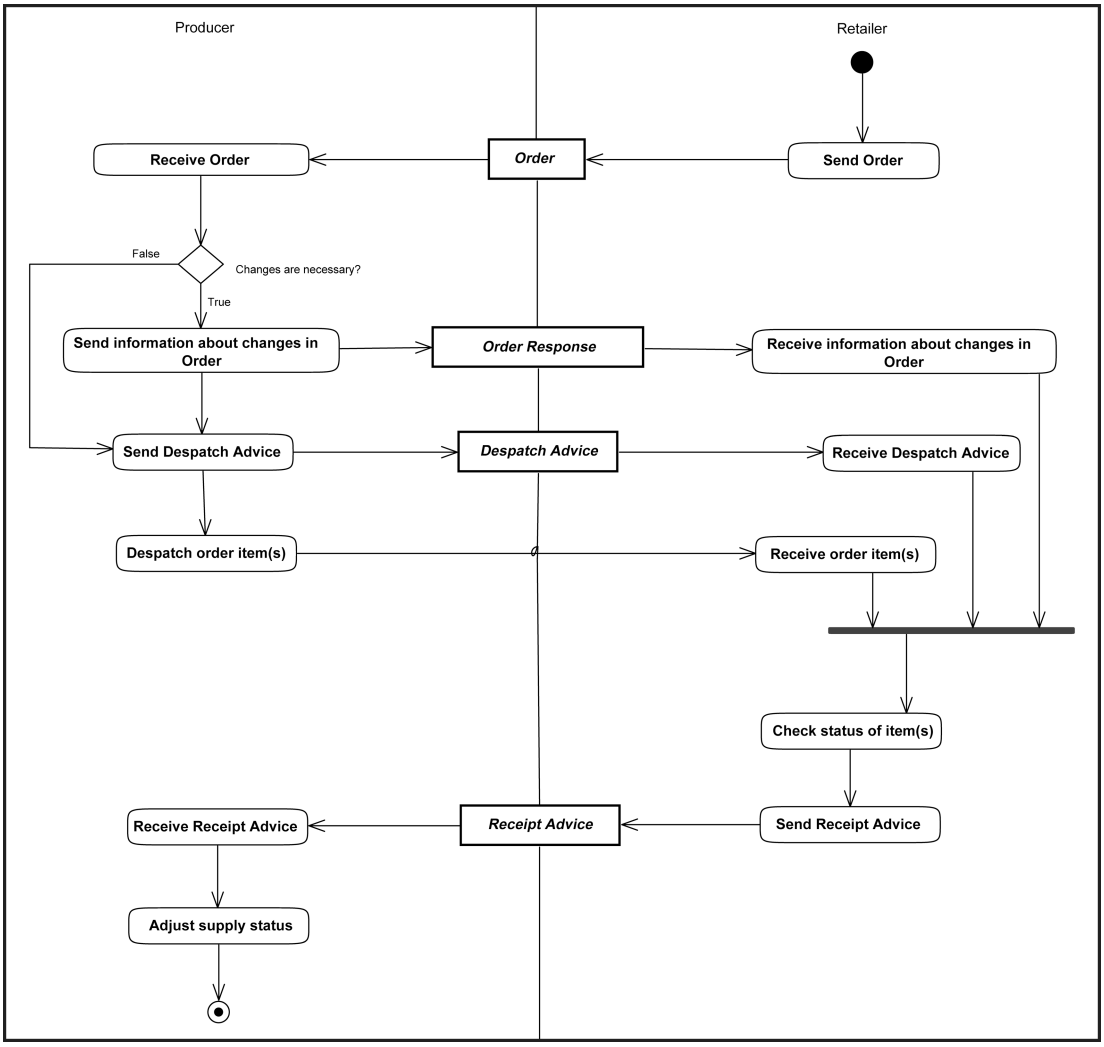

Figure 54 — Initial Stocking of the Area by Producer and Retailer

5.3.3.5.4.5 Periodic Replenishment

Periodically, the retailer's system calculates the quantities needed for replenishment of the area. From the result, an order is sent, and the producer responds with a direct delivery within 48 hours.

The replenishment process uses the same documents in the same order as the Initial Stocking process, so the duplicate diagram is omitted here; see 5.3.3.5.4.4 Initial Stocking of the Area by Producer and Retailer. It must be remembered, however, that the two processes are taking place at different points in time, so their pre and post conditions will be different.

5.3.3.5.4.6 Report of Sales and Inventory Movement on Customer Demand

Sales and inventory movement information is transferred daily from the retailer to the producer.

The process for sales and inventory reporting is the same as in CRP (see Figure 48).

5.3.3.5.4.7 Invoicing for Replenishment On Customer Demand

An invoice is sent either on a delivery or a sales basis.

The invoice process for Replenishment On Customer Demand is the same as for CRP (see Figure 49).

5.3.3.5.4.8 Synchronizing Stock Information

Information about the actual stock is synchronised periodically (for example, every one to three months). Synchronization occurs at least once a year together with a physical inventory.

The stock synchronization process for Replenishment On Customer Demand is the same as in CRP (see Figure 50).

5.3.3.5.4.9 Changes to the Article Catalogue

In the event of a change, either inside an item belonging to the Catalogue or the relationship of an item to the Catalogue, information about the change is sent to the retailer by sending an updated Catalogue document. Item change is indicated by an optional Action Code field in each changed Catalogue Line.

The process for changing the catalogue in Replenishment On Customer Demand is the same as in CRP (see Figure 51).

5.3.4 Make

The make processes include, production activities, packaging, staging product, and releasing. It also includes managing the production network, equipment and facilities, and transportation.

As these are traditionally internal organizational activities they are not included in this release. However we anticipate and welcome submissions from the industry for document types that may be utilized in these processes.

5.3.5 Deliver

5.3.5.1 Logistics

5.3.5.1.1 Fulfilment Introduction

Fulfilment is the collaboration in which the goods or services are transferred from the Despatch Party to the Delivery Party.

Document types in these processes are Despatch Advice, Receipt Advice, Order Cancellation, Order Change, and Fulfilment Cancellation.

In common practice, fulfilment is either supported by a proactive Despatch Advice from the Despatch Party or by a reactive Receipt Advice from the Delivery Party.

If the Customer is not satisfied with the goods or services, they may then cancel or change the order (see 5.3.3.4 Ordering (post-award)). The Seller may have a fulfilment (or customer) service dealing with anomalies.

Cancellation of a Despatch Advice or Receipt Advice is accomplished using the Fulfilment Cancellation document (see 5.3.5.1.4 Fulfilment Cancellation Business Rules).

5.3.5.1.2 Despatch Advice Business Rules

The Despatch Advice is sent by the Despatch Party to the Delivery Party to confirm shipment of items.

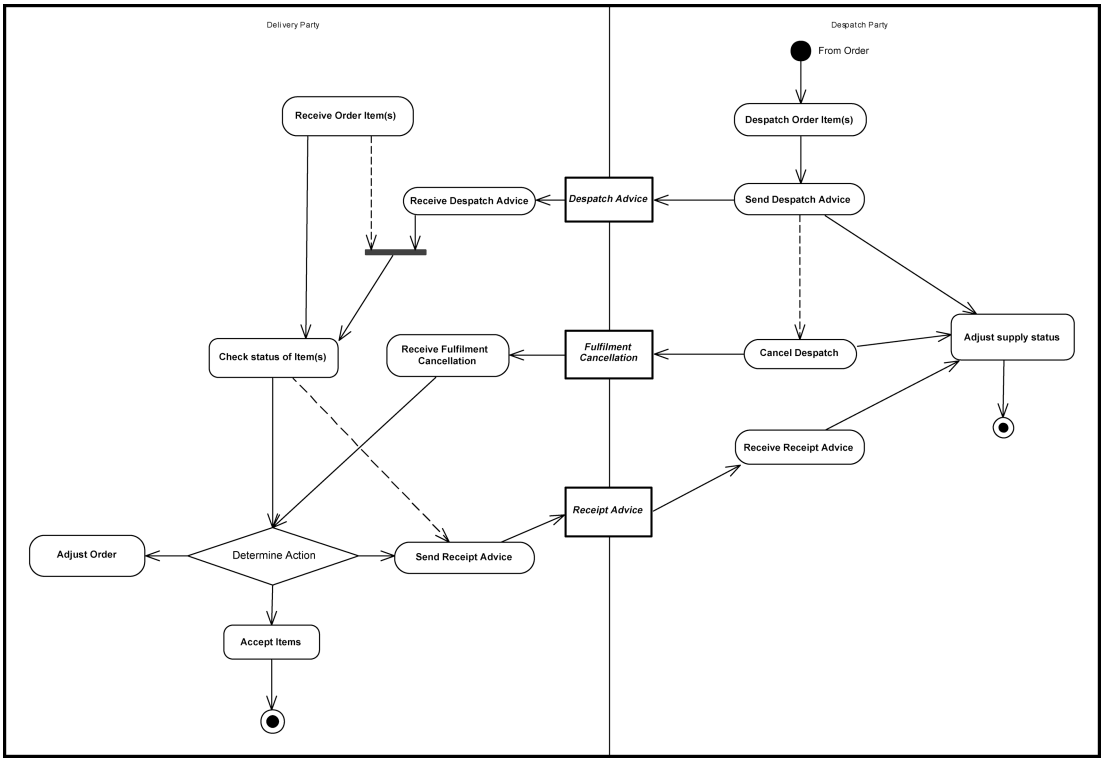

Figure 55 — Fulfilment with Despatch Advice

The Despatch Advice provides for two situations:

1) Organization of the delivery set of items by Transport Handling Unit(s) so that the Receiver can check the Transport Handling Unit and then the contained items. Quantities of the same item on the same Order Line may be separated into different Transport Handling Units and hence appear on separate Despatch Lines within a Transport Handling Unit.

2) Organization of the delivery set of items by Despatch Line, annotated by the Transport Handling Unit in which they are placed, to facilitate checking against the Order. For convenience, any Order Line split over multiple Transport Handling Units will result in a Despatch Line for each Transport Handling Unit they are contained in.

Additionally, in either case, the Despatch Advice may advise:

— Full Despatch—advising the Recipient and/or Buyer that all the items on the order will be, or are being, delivered in one complete consignment on a given date.

— Partial Despatch—advising the Recipient and/or Buyer that the items on the order will be, or are being, partially delivered in a consignment on a given date.

Despatch Lines of the Despatch Advice need not correspond one-to-one with Order Lines, and are linked by a reference. The information structure of the Despatch Advice may result in multiple Despatch Lines from one Order Line. Equally, partial despatch may result in some Order Lines not being matched by any Line in a Despatch Advice.

Within a Despatch Advice, an Item may also indicate the Country of Origin and the Hazardous nature of the Item.

5.3.5.1.3 Receipt Advice Business Rules

The Receipt Advice is sent by the Delivery Party to the Despatch Party to confirm receipt of items. It also is capable of reporting shortages or damaged items.

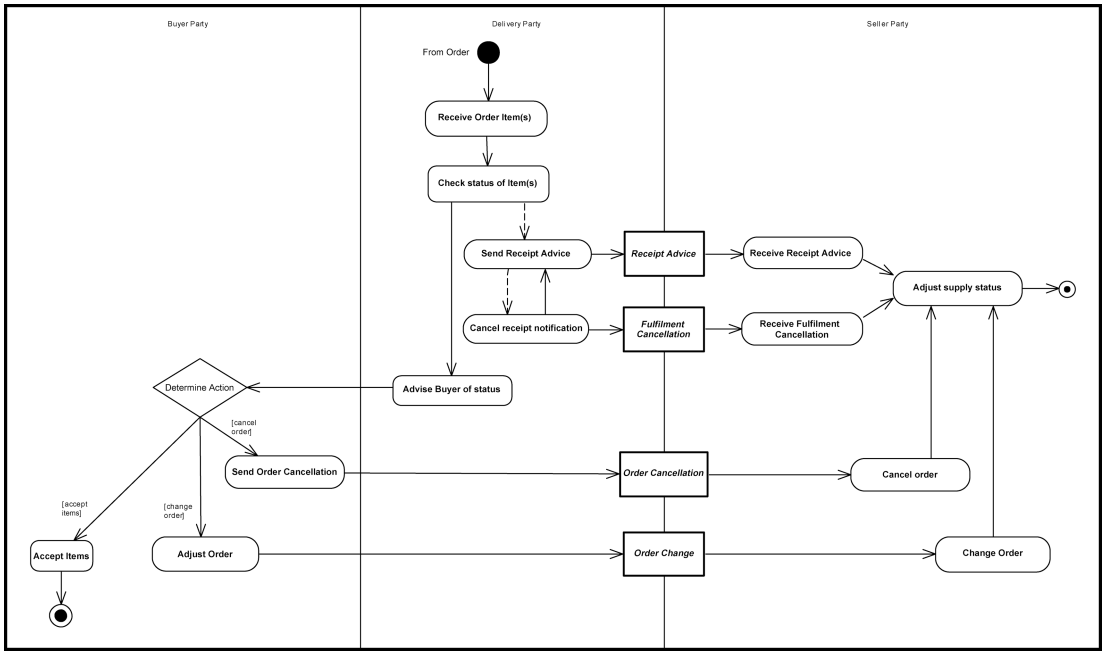

Figure 56 — Fulfilment with Receipt Advice

The Receipt Advice provides for two situations. For ease of processing claimed receipt against claimed delivery, it must be organised in the same way as the corresponding Despatch Advice:

1) Indication of receipt by Transport Handling Unit(s) and contained Receipt Lines one-to-one with the Despatch Advice as detailed by the Seller party, or

2) Indication of receipt by Receipt Lines annotated by Transport Handling Unit, one-to-one with the Despatch Advice as detailed by the Seller party.

The Receipt Advice allows the Delivery Party to state any shortages from the claimed despatch quantity and to state any quantities rejected for a given reason.

5.3.5.1.4 Fulfilment Cancellation Business Rules

In real life, the sender of a Despatch Advice or Receipt Advice sometimes needs to cancel the document after it has been sent. The Fulfilment Cancellation document is provided for this purpose.

For example, a Despatch Advice may later be cancelled by the Supplier when a problem with shipment prevents the delivery of goods, or the goods to be shipped are not available, or the order is cancelled; in these cases, the customer cancels receipt and adjusts the order accordingly (see Figure 55).

Similarly, a Receipt Advice may later be cancelled by the customer (see Figure 56) if the customer discovers an error in ordering (failure to follow formal contractual obligations, incorrect product identification, etc.) or a problem with a delivered item (malfunction, missing part, etc.). In this case, the billing and payment process may be put on hold.

5.3.5.2 Transport

5.3.5.2.1 International Freight Management Introduction

Freight management for domestic trade is typically accomplished using Despatch Advice and Receipt Advice (see 5.3.5.1 Logistics). The additional processes shown in Figure 57 are engineered to support the ordering and management of logistical services for international trade.

With receipt of an order and acknowledgement by the Supplier Party that the goods are available and ready to be shipped, the Consignor or Consignee initiates the transportation arrangements. This includes booking the consignment with a

Transport Service Provider such as the Freight Forwarder or Carrier and advising the Delivery Party of the arrangements as needed.

Document types in these processes are Forwarding Instructions, Packing List, Bill Of Lading, and Waybill. (Regarding the Transportation Status document type, see 5.3.5.3 Freight Status Reporting).

It should be noted that these processes involve the Consignee and Consignor and do not cover all the logistical processes required to physically move the goods or regulatory notifications such as Customs declarations.

NOTE For a discussion of the difference between *consignment* (consignor to consignee) and *shipment* (shipper to recipient), see 5.2.12 Shipment vs. Consignment.

For a discussion of the difference between *transport* and *transportation*, see 5.2.13 Transport vs. Transportation.

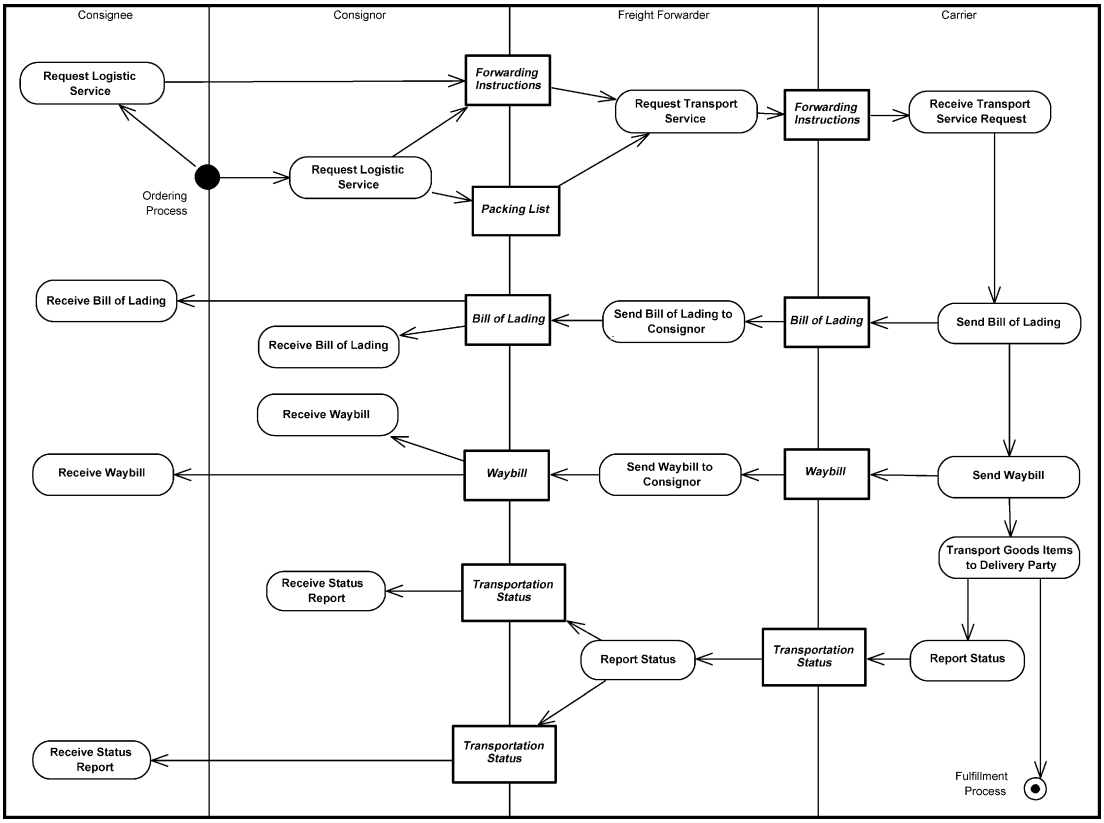

Figure 57 — Initiate Freight Management Process

5.3.5.2.2 Forwarding Instructions

Forwarding Instructions are normally used by any party who gives instructions for the transportation services required for a consignment of goods (the Transport Service Buyer) to any party who is contracted to provide the transportation services (called the Transport Service Provider). Forwarding Instructions may also be used by any party who requests a booking of shipment space to be made for the transportation services required for a consignment of goods to any party who will provide the underlying transportation services. The parties who issue this document are commonly referred to as the shipper, consignee, or consignor, while the parties who receive this document are forwarders, carriers, shipping agents, etc.

Forwarding Instructions may also be issued by a freight forwarder or shipping agent in their capacity as a Transport Service Buyer. This document may be used to arrange for the transportation:

— Of different types of goods or cargoes

— Whether containerized or non-containerized

— Through different modes of transport, and

— From any origin to any destination.

5.3.5.2.3 Packing List

A Packing List is normally issued by the Consignor. It states the distribution of goods in individual packages.

5.3.5.2.4 Bill of Lading

A Bill Of Lading is a transport document that is the evidence of a contractual agreement between the parties for the transportation service. The document evidences a contract of carriage by sea and the acceptance of responsibility for the goods by the carrier, by which the carrier undertakes to deliver the goods against surrender of the document. It is in common use for ocean or inland waterways modes of transport. The Bill of Lading (B/L) may serve as a document of title. A provision in the document that the goods are to be delivered to the order of a named person, or to order, or to bearer, constitutes such an undertaking.

A Bill of Lading is normally issued by the party who provides the physical transportation services (e.g., the maritime carrier) to the party who gives instructions for the transportation services (shipper, consignor, etc.) as a receipt for the cargo and sometimes of instructions, stating the details of the transportation, charges, and terms and conditions under which the transportation service is provided.

A Bill of Lading may also be issued by the party who acts as an agent for the carrier or other agents to the party who gives instructions for the transportation services (shipper, consignor, etc.) stating the details of the transportation, charges, and terms and conditions under which the transportation service is provided, but who does not provide the physical transportation service. In such case a Bill of Lading is signed “as agent”.

Much of the information contained in the Bill of Lading corresponds to the information on the Forwarding Instructions.

A freight forwarder, who can be either a Transport Service Provider or a Transport Service User according to different circumstances and depending on the contractual interlocutor, can assume responsibility for the shipment with regards to the shipper and issue Bills of Lading as a common carrier, a contractual carrier, or as a Non Vessel Operating Common Carrier (NVOCC). In such case, when the transportation is multimodal, it can provide a multimodal Bill of Lading.

5.3.5.2.5 Waybill

A Waybill is a transport document issued by the party who undertakes to provide transportation services, or undertakes to arrange for their provision, to the party who gives instructions for the transportation services (shipper, consignor, etc.). It states the instructions for the beneficiary and may contain the details of the transportation, charges, and terms and conditions under which the transportation service is provided.

Unlike a Bill Of Lading, a Waybill is not negotiable and cannot be assigned to a third party (endorsement). It may be issued as a cargo receipt and is not required to be surrendered at the destination in order to pick up the cargo. This may simplify the documentation procedures between a Transport Service Buyer and a Transport Service Provider, but using this document in combination with international payments (e.g., documentary credits) is not advisable.

A freight forwarder may decide to issue a waybill to communicate consignment, transport, and conveyance information to third parties, be they shippers, subcontractors, transport operators, or authorities.

5.3.5.2.6 Weight Statement

A Weight Statement is a transport document verifying the declared true gross mass of a packed container. Working with this knowledge avoids injury, container loss, damage to cargo, etc. Formally verifying the gross mass may be a condition for transport.

5.3.5.3 Freight Status Reporting

Freight Status Reporting is the process by which a Transport Service Provider (such as a Carrier or Freight Forwarder) communicates the status of shipments currently under their management to the Transport Users (such as a Freight Forwarder, Consignee, or Consignor).

A Transportation Status document is provided either through a Transportation Status Request document or through an agreed status reporting procedure.

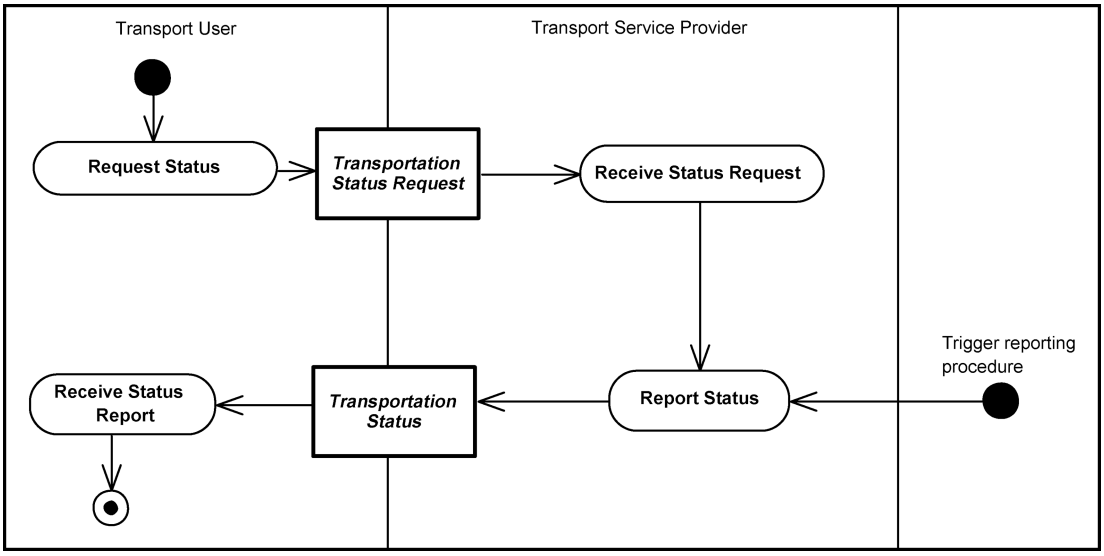

Figure 58 — Freight Status Reporting Process

5.3.5.4 Certification of Origin of Goods

When a Consignor exports certain goods they may be required to attest to the origin of the goods. A Certificate Of Origin is a document required by regulatory bodies declaring that goods in a particular international shipment are of a certain origin.

It is the responsibility of the Exporter to sign the Certificate of Origin Application document and submit it for authentication to a recognized authority (such as a local chamber of commerce or designated government agency or board). This party becomes the Endorser and will issue the Certificate of Origin document. To do this the Endorser must have access to other documents, such as the commercial Invoice and Bill Of Lading, in order to verify the Exporter's claims that the goods originated in that country. In effect, the Certificate of Origin document is a dossier describing a set of related documents. After it is issued, the Certificate of Origin is sent to the Importer.

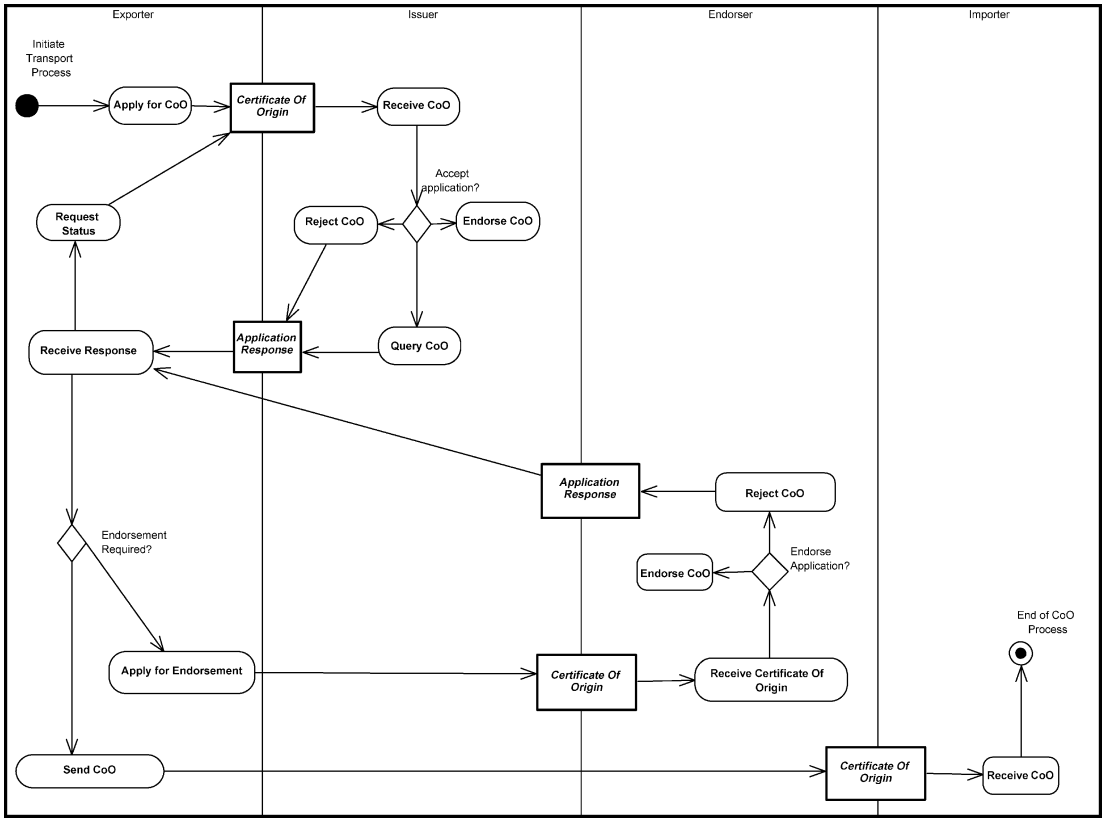

Figure 59 — Certification of Origin of Goods Process

5.3.5.5 Cross Border Regulatory Reporting

The major applications for Cross Border Regulatory reporting are:

— Single Window Systems

— Co-ordinated Border Management

— Data Re-use

— Supply Chain Security

— Security Filing

— Trade & Transport Data Pipelines

— Trade Data Intelligence

Work is currently in progress within the UBL Technical Committee to develop UBL documents that work with the cross border regulatory requirements. These will provide a link between the information contained in commercial business documents and the information required for reporting to customs and other government agencies for the clearance of goods, cargo and means of transport. These UBL documents will complement the WCO Data Model standards.

5.3.5.6 Intermodal Freight Management

5.3.5.6.1 Intermodal Freight Management Introduction

Intermodal transport implies the use of a combination of transport modes. Any support for the management of such chains has to support the modal change of cargo flows from one mode to another in order to create seamless sequences of transportation legs. Quite often the end legs are carried by road, but there are instances of short sea shipping, inland waterways, and rail being used as end legs.

The Intermodal Freight Management process differs from conventional international freight management in that it may involve multiple different transport modes. The focus is the multimodal transport chain as seen from the Transport User's point of view. The Transport User needs information about all the possible transport services that can be used to build a complete transport chain. If the choices to be made by the Transport User or his agent are based upon the qualities of the transport services themselves, and not by which transport mode is used, the description of the transport services and the exchanges of information about the transport roles and services must be simple and common. Taking an intermodal approach requires a generalized view of the business processes, parties, and roles involved in the process.

The roles of the various Parties are defined as follows:

— The *Transport User* is the role representing anyone who needs to have cargo transported. The Transport User provides the Transport Service Provider with instructions and detailed information about the cargo to be transported.

— The *Transport Service Provider* is the role that ensures the transport of the cargo from the origin to the destination. This includes the management of the transport services and the operation of the transport means and handling equipment. A Transport Service Provider may also provide administrative services required for moving the cargo, such as cargo inspection.

— The *Transportation Network Manager* is the role that extracts all information available regarding the infrastructure related to planning and executing transport and makes this information available to the Transport Service Provider.

— The *Transport Regulator* is the role that receives all mandatory reporting (and checks if reporting has been carried out) in order to ensure that all transport services are completed according to existing rules and regulations.

It should be noted that one Party (person or organization) may take on different roles. For example, a freight forwarder is, on the one hand, a Transport Service Provider when its client is a Transport User. On the other hand, the freight forwarder is a Transport User when it acquires services from subcontractors to ensure that a transport service is carried out between origin and destination. In so doing, the freight forwarder can operate as agent, thus arranging a contractual relationship between the carrier and the shipper, or as principal, thus organizing the transportation chain by concluding contracts in its own name on behalf of the shipper(s).

The Intermodal Freight Management process takes place in three stages:

— *Planning*: In this stage, the Transport Users express their transport demand in a standard format, the Transport Service Description Request. Transport Service Providers plan their transport services and announce them to Transport Users using the Transport Service Description. This stage also covers the arrangement of transport services between Transport Users and Transport Service Providers, establishing Transport Execution Plans. Once a Transport Execution Plan has been established, a Goods Item Itinerary is sent from the Transport Service Provider to the Transport User. The Goods Item Itinerary provides additional information related to the complete transport service.

— *Execution*: In this stage, Transport Service Providers perform the physical transport of the cargo, and they exchange information related to the status of the transported cargo with the Transport Users using the Transportation Status document. Furthermore, in this stage Transport Service Providers exchange regulatory information with Transport Regulators as well as receive status regarding the transport infrastructure from Transportation Network Managers using the Transport Progress Status document.

— *Completion*: This stage facilitates the issuing of proofs of delivery, claims, and invoices between Transport Service Providers and Transport Users.

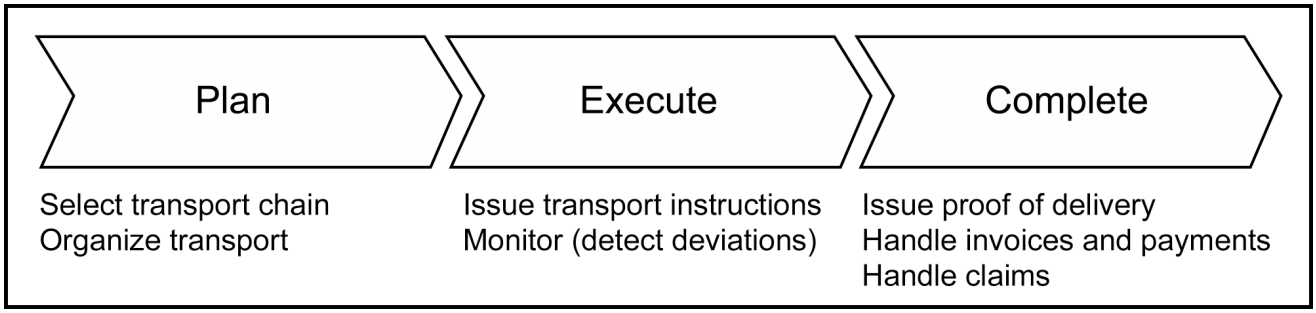

Figure 60 — The Generic Freight Management Process

These three stages are detailed in the following diagram, which shows the part played in the Intermodal Freight Management process by the UBL document types Transport Service Description, Transport Service Description Request, Transport Execution Plan, Transport Execution Plan Request, Transportation Status, Transportation Status Request, Transport Progress Status, Transport Progress Status Request, Goods Item Itinerary, and Freight Invoice.

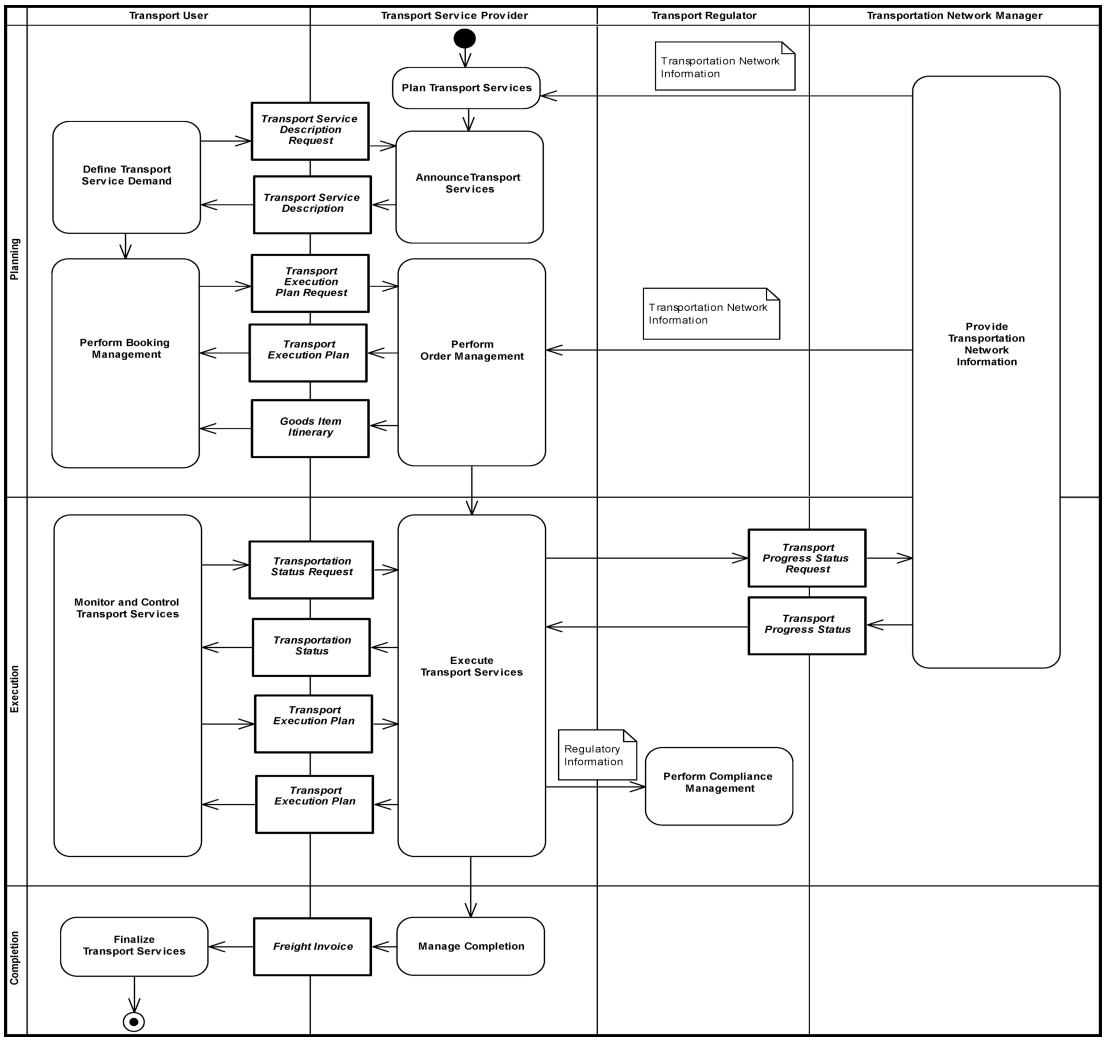

Figure 61 — The Intermodal Freight Management Process

5.3.5.6.2 Announcing Intermodal Transport Services

The Transport Service Description is used to publish information about a transport service. A Transport Service Description Request is used to request such information. A transport service can be the physical transport of cargo between an origin and a destination, and it can also refer to other transport-related services such as terminal services, warehousing services, handling services, or document handling services.

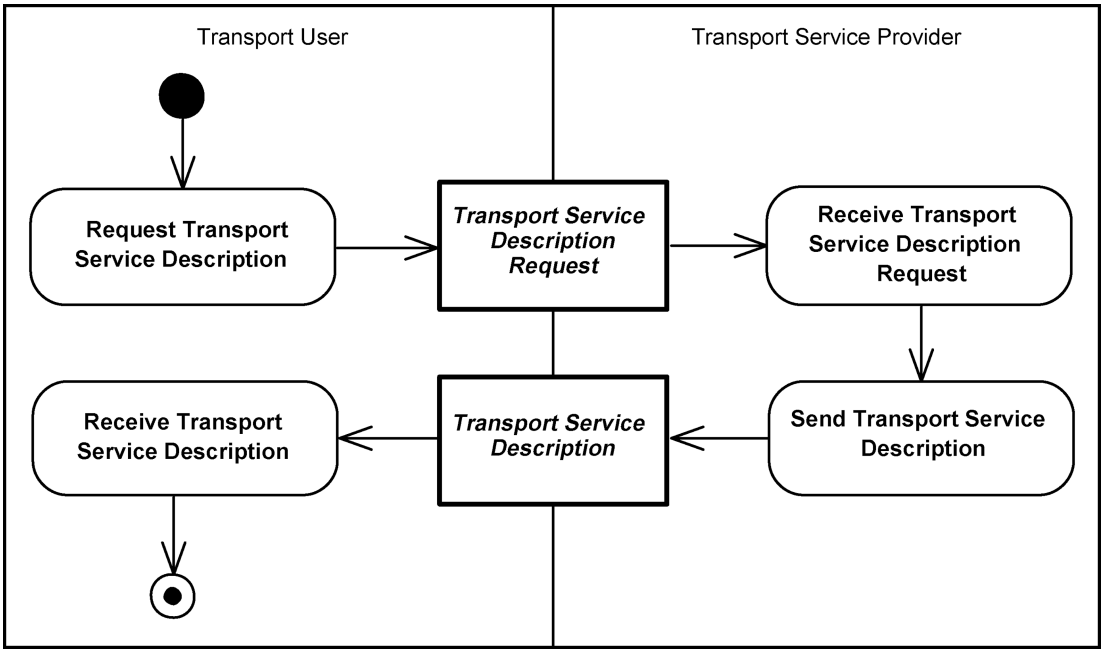

Figure 62 — Transport Service Description

5.3.5.6.3 Establishing a Transport Execution Plan

The Transport Execution Plan is a plan established between a Transport User and a Transport Service Provider in order to collaborate and document the details surrounding the provision of a required transport service. Depending on the nature of the transport service and the business relationship between the Transport User and the Transport Service Provider, the process of establishing a Transport Execution Plan may be carried out by means of multiple interactions between the two roles, from the initial request from the Transport User up to the final agreement of the Transport Execution Plan among the parties involved.

The following diagram (Figure 63) shows the message exchange involved in a basic scenario. A Transport Execution Plan Request is sent from the Transport User in order to request a transport service. If the Transport Service Provider accepts the transport service request, he responds with a confirmed Transport Execution Plan. If the Transport Service Provider does not accept the transport service request, he responds with a rejected Transport Execution Plan.

The handling of a Transport Service Request will in many cases depend upon whether or not there is a pre-established agreement between the Transport User and the Transport Service Provider. If there is a pre-established agreement, the Transport Service Request can typically be considered a call-off from the agreement between the two parties. (An established framework agreement or contract usually defines terms and conditions and a total capacity limit, e.g., 100 container spaces on a vessel per year. A call-off occurs when the Transport User places an order against this agreement, for example a booking of 10 of the 100 container spaces.) The Transport User can confirm the Transport Execution Plan Request without the need to make a careful examination of the Transport Execution Plan submitted by the Transport Service Provider. The Transport User then sends a Transport Execution Plan with a status code indicating confirmation. Assuming acceptance by the Transport Service Provider, this scenario is considered a two-step choreography.

If a pre-established agreement does not exist (e.g., spot market services), the Transport User issues a Transport Execution Plan Request with a status code indicating that the Transport Execution Plan is not yet confirmed. The Transport User only confirms the Transport Execution Plan after a careful analysis of what has been submitted by the Transport Service Provider. This scenario is a three-step choreography where the Transport User confirms the Transport Execution Plan content in his second or subsequent response to the Transport Service Provider.

Updates to the Transport Execution Plan may be issued by either the Transport User or the Transport Service Provider. If the Transport User wants to update an existing Transport Execution Plan, a new instance of a Transport Execution Plan must

be issued with reference to the original Transport Execution Plan. Similarly, if the Transport Service Provider wants to update an existing Transport Execution Plan, a new Transport Execution Plan replaces the original Transport Execution Plan with a reference to the original one. In either case, the Transport Execution Plan must include a document status code indicating that this is an update of the original content.

The cancellation of a Transport Execution Plan may be requested by either the Transport Service Provider or the Transport User. In either case, the Transport Execution Plan is sent with a document status code indicating that the Transport Execution Plan should be cancelled. For the cancellation to be effective, it must be accepted by the party receiving the cancellation request. Acceptance is signified by sending back a Transport Execution Plan with a status code indicating confirmation; rejection of the cancellation is signified by issuing a Transport Execution Plan with a status code indicating rejection.

Upon completion of the transport service covered by the Transport Execution Plan, a final Transport Execution Plan document is sent from the Transport Service Provider to the Transport User that includes a document status code indicating that the transport service is completed.

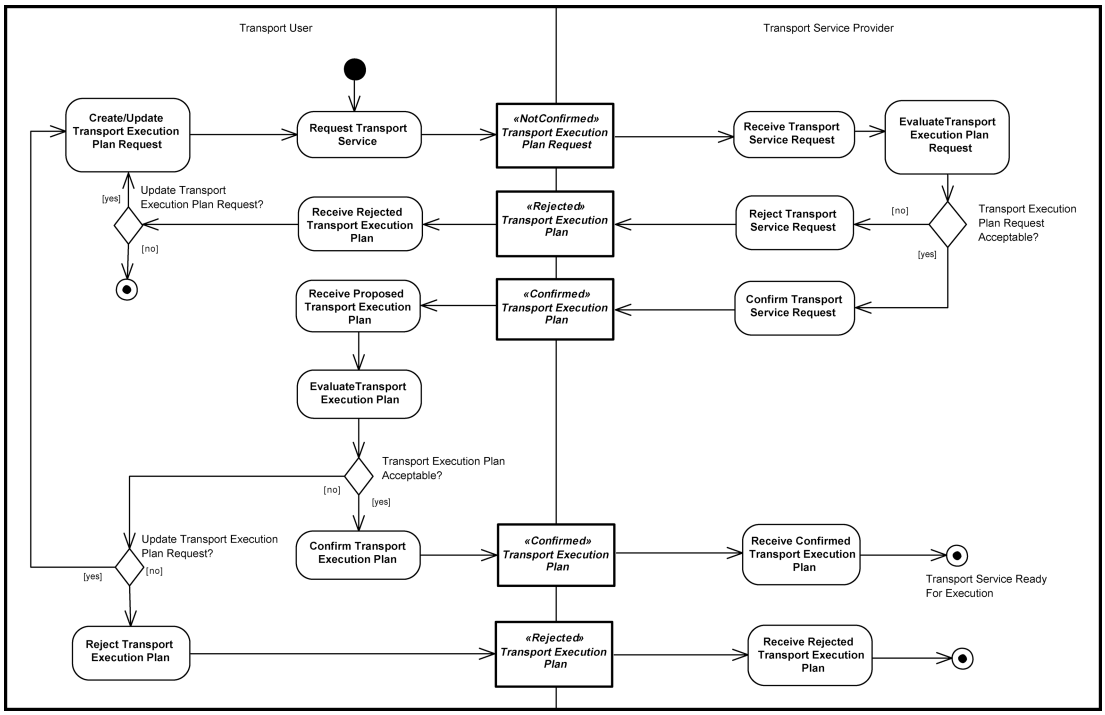

Figure 63 — Transport Execution Plan

5.3.5.6.4 Providing an Itinerary for a Transport Service

The Goods Item Itinerary specifies the route and time schedule for one or more transported items and is issued from the Transport Service Provider to the Transport User. The Goods Item Itinerary is initially issued from the Transport Service Provider to the Transport User after a Transport Execution Plan is confirmed by both parties. It may contain one or more transport segments with different Transport Execution Plans employing different Transport Service Providers. One transport service (one Transport Execution Plan) may cover more than one segment (leg).

In addition to providing an overview of the initial route and time schedule, the Goods Item Itinerary is used to document progress by recording new estimated times for departure or arrival and actual departure and arrival times. So when updates to the initial transport execution schedule occur, a new version of the Goods Item Itinerary is issued to the Transport User. A Goods Item Itinerary document thus contains information that may be used for analyzing the performance (in time) of transport services and for tracing the progress of cargo in transit if such analysis is required.

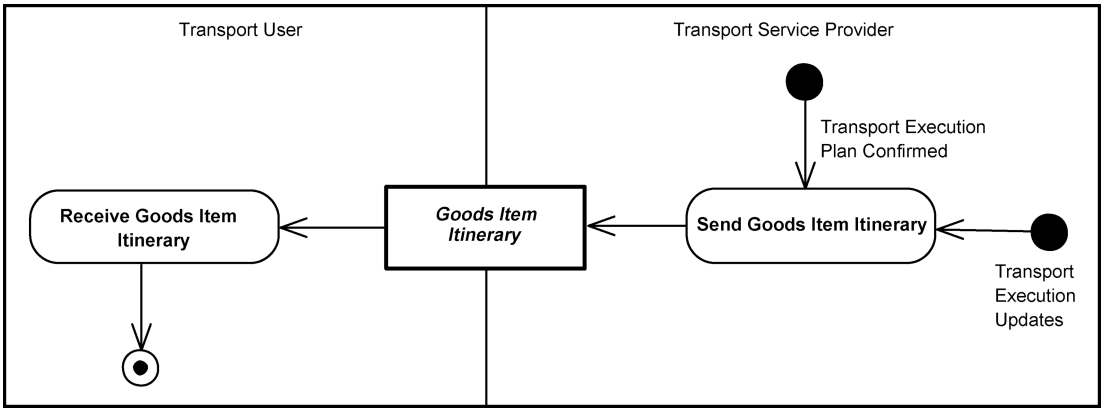

Figure 64 — Goods Item Itinerary

5.3.5.6.5 Reporting Transport Means Progress Status

The Transport Progress Status collects and reports information about the status of the transport means. The Transport Service Provider issues a Transport Progress Status Request to ask the Transportation Network Manager for status information related to a specific transport vehicle, using the vehicle identification number.

The Transportation Network Manager then provides information about the location and time schedule status to the Transport Service Provider. During a transport service, there might be a number of information providers taking on the Transportation Network Manager role, offering Transport Progress Statuses to the Transport Service Provider.

The most typical use of Transport Progress Status is to ask assistance from the Transportation Network Manager when estimated times of arrival are established. Reporting on the status of the goods themselves is covered by the Freight Status Reporting process (see 5.3.5.3 Freight Status Reporting).

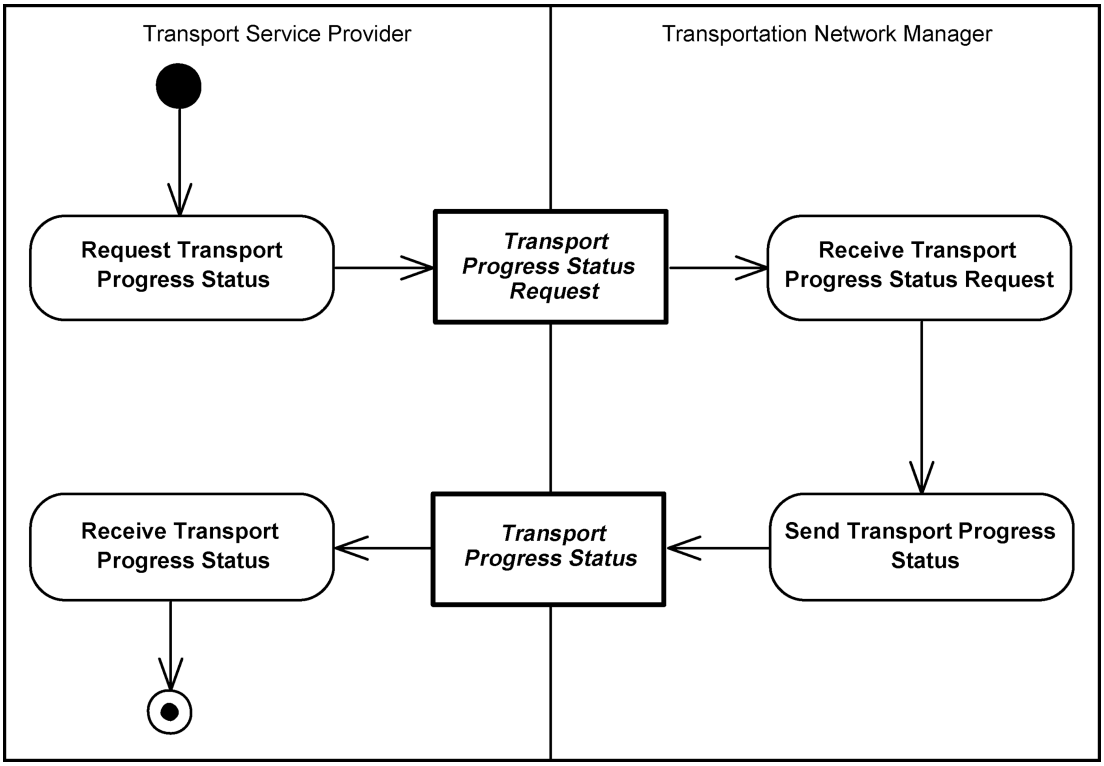

Figure 65 — Transport Progress Status

5.3.6 Return

Organizations may be required to handle the return of containers, packaging, or defective product. The return involves the management of business rules, return inventory, assets, transportation, and regulatory requirements.

Currently there are no specific UBL digital business documents associated with these processes. However we anticipate and welcome submissions from the industry for document types that may be utilized in these processes.

5.3.7 Pay

5.3.7.1 Billing

5.3.7.1.1 Billing Introduction

In the Billing process, a request is made for payment for goods or services that have been ordered, received, or consumed. In practice, there are several ways in which goods or services may be billed.

Document types in these processes are Invoice, Credit Note, Debit Note, and Application Response.

For UBL we assume the following billing methods:

1) Traditional Billing
    a) Using Credit Note
    b) Using Debit Note
2) Self Billing (also known as billing on receipt)
    a) Using Credit Note
    b) Using Self Billed Credit Note

5.3.7.1.2 Billing Business Rules

An Invoice defines the financial consequences of a business transaction. The Invoice is normally issued on the basis of one despatch event triggering one Invoice. An Invoice may also be issued for pre-payment on a whole or partial basis. The possibilities are:

— Prepayment invoice (payment expected)

— Proforma invoice (pre Despatch Advice, payment not expected)

— Normal Invoice, on despatch for despatched items

— Invoice after return of Receipt Advice

The Invoice only contains the information that is necessary for invoicing purposes. It does not reiterate any information already established in the Order, Order Change, Order Response, Despatch Advice, or Receipt Advice that is not necessary when invoicing. If necessary, the Invoice refers to the Order, Despatch Advice, or Receipt Advice by a Reference for those documents.

The Invoice allows for compound taxes, the sequence of calculation being implied by the sequence of information repeated in the data stream (e.g., Energy tax, with VAT—Value Added Tax—superimposed).

Charges may be specified either as a lump sum or by percentage applied to the whole Invoice value prior to calculation of taxes. Such charges cover:

— Packaging

— Delivery/postage

— Freight

— Documentation

Each Invoice Line refers to any related Order Line(s) and may also refer to the Despatch Line and/or Receipt Line.

5.3.7.1.3 Traditional Billing

5.3.7.1.3.1 Traditional Billing Introduction

Traditional billing is where the supplier invoices the customer when the goods are delivered or the services are provided. In this case, the invoice may be created at the time of despatch or when the Delivery Party acknowledges that the goods have been received (using a Receipt Advice).

When there are discrepancies between the Despatch Advice, Receipt Advice, or Invoice and the goods actually received, or the goods are rejected for quality reasons, the customer may send an Application Response or a Debit Note to the supplier. The supplier may then issue a Credit Note or another Invoice as required.

A Credit Note or Debit Note may also be issued in the case of retrospective price change.

Credit Notes or Debit Notes may be also issued after the Billing collaboration (as part of the Payment collaboration).

5.3.7.1.3.2 Billing Using Credit Notes

Billing using Credit Note is shown in the following diagram.

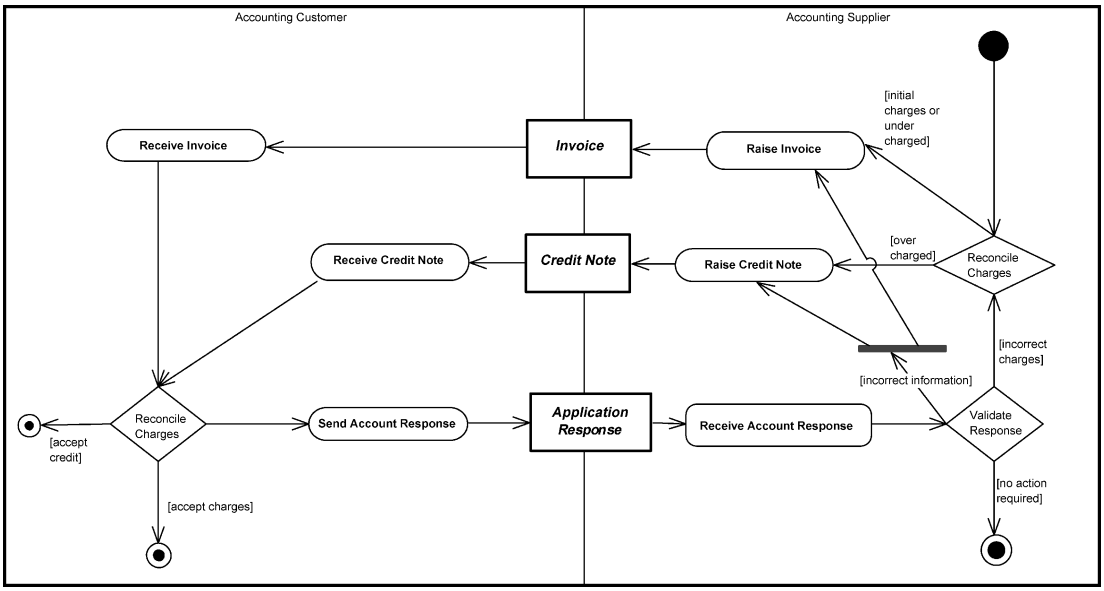

Figure 66 — Billing with Credit Note Process

When using Credit Notes, the Supplier (in their Accounting role) is responsible for specifying the tax requirements.

5.3.7.1.3.3 Billing Using Debit Notes

Billing using Debit Note is shown in the following diagram.

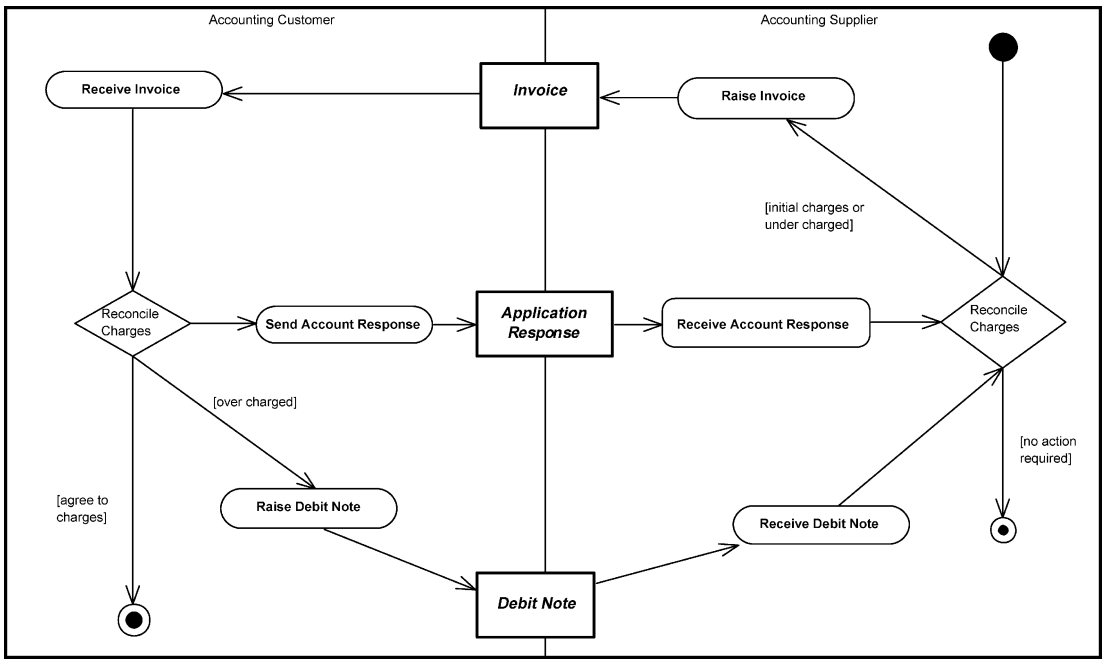

Figure 67 — Billing with Debit Note Process

When using Debit Notes, both the Supplier (in their Accounting role) and the Customer (in their Accounting role) are responsible for providing taxation information.

5.3.7.1.4 Self Billing

5.3.7.1.4.1 Self Billing Introduction

A self billing process is where a Customer “invoices” itself, *in the name and on behalf of* the Supplier, and provides the Supplier with a copy of the self billed invoice.

5.3.7.1.4.2 Self Billing Using Credit Notes

Self Billing using Credit Note is shown in the following diagram.

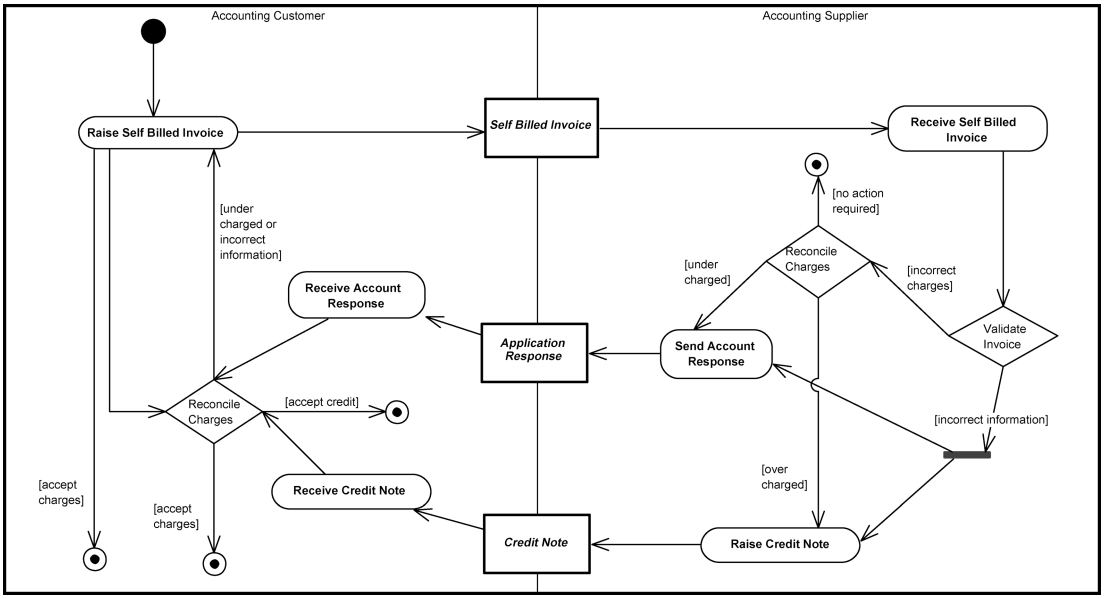

Figure 68 — Self Billing with Credit Note Process

If the Supplier finds that the Self Billed Invoice is incorrect, e.g., wrong quantities or wrong prices, or if the goods have not been invoiced at all, it may send an Application Response or a Credit Note to the Customer. The customer may then verify whether the adjustment is acceptable or not and consequently issue another Self Billed Invoice or a Self Billed Credit Note.

5.3.7.1.4.3 Self Billing Using Self Billed Credit Notes

Self Billing using Self Billed Credit Note is shown in the following diagram.

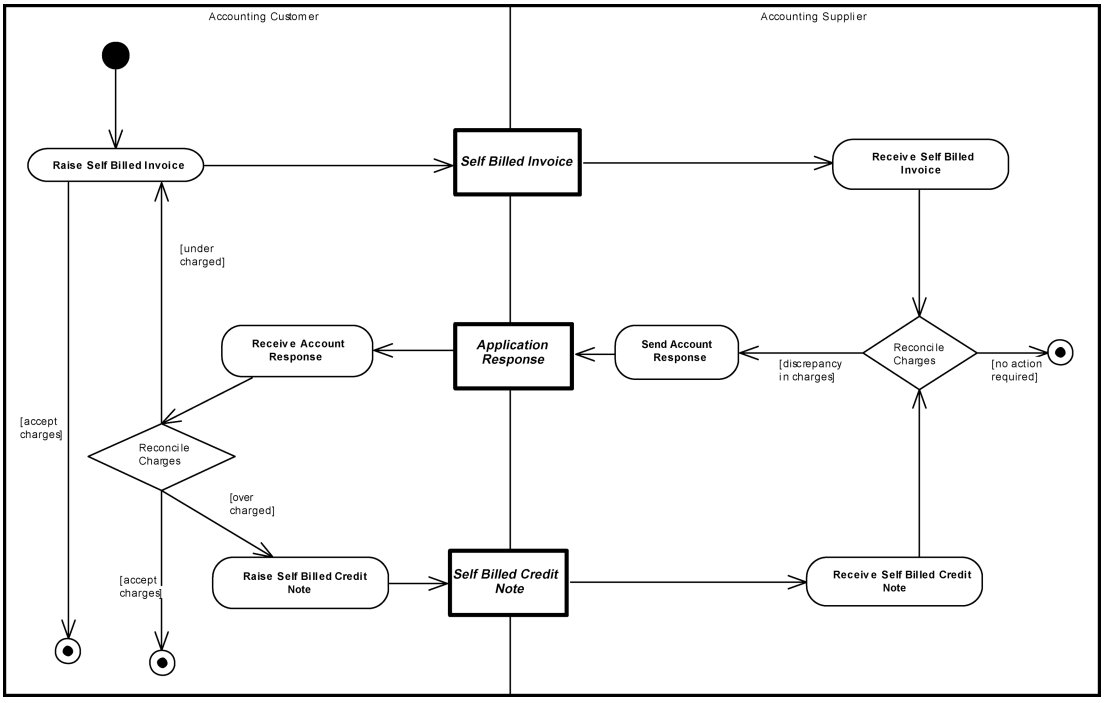

Figure 69 — Self Billing with Self Billed Credit Note Process

When using Self Billed Credit Notes, the Customer is raising the Self Billed Credit Note *in the name and on behalf of* the Supplier. Therefore the Supplier and the Customer are still both responsible for providing taxation information.

5.3.7.1.5 Reminder for Payment

A Reminder may be used to notify the Customer of accounts due to be paid.

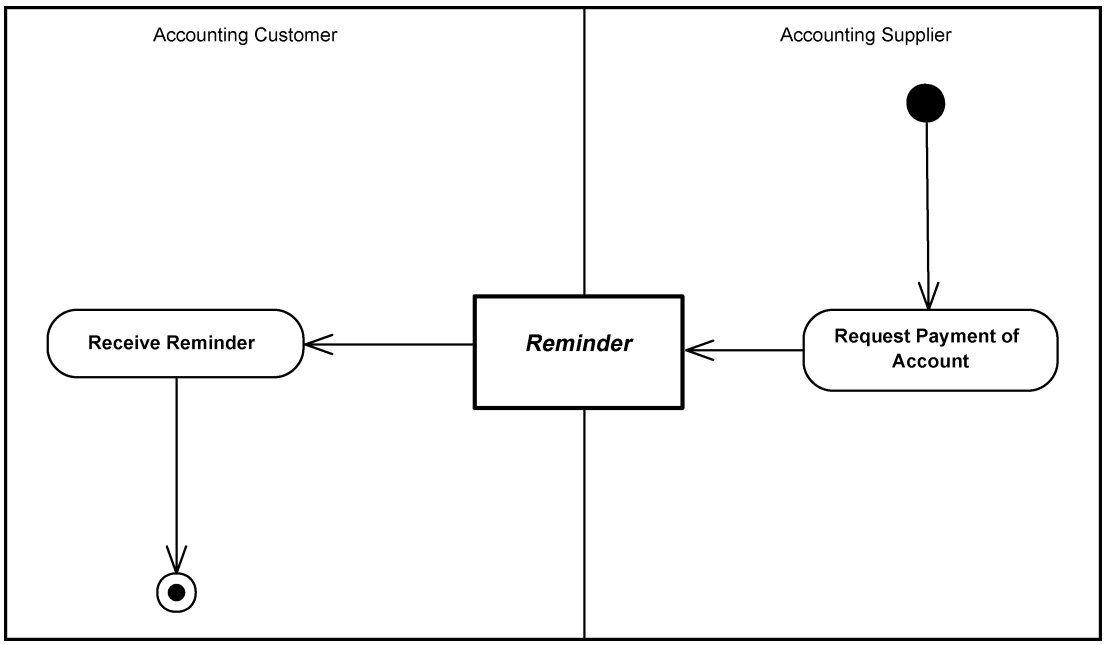

Figure 70 — Reminder for Payment Process

5.3.7.2 Freight Billing

An extension of the Billing process is that of Freight Billing. This represents the billing process between the Transport Service Buyer (e.g., the Consignor) and Transport Service Provider (e.g., a Freight Forwarder) through the use of an invoice for freight charges.

The Transport Service Provider initiates the process of billing the Transport Service Buyer for logistic services.

The Freight Invoice lists the charges incurred in order to fulfil the agreed service.

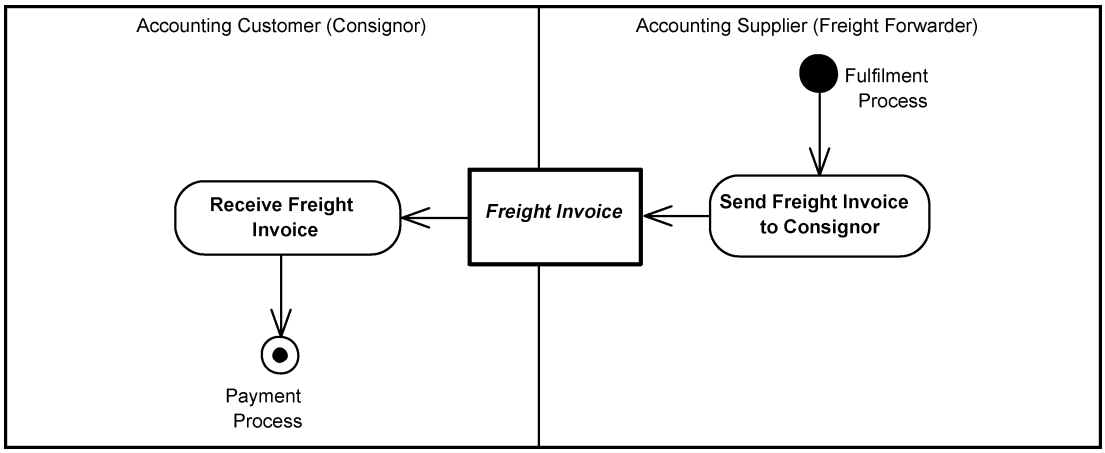

Figure 71 — Freight Billing Process

5.3.7.3 Utility Billing

This process defines the billing process for invoicing between suppliers of utilities (including electricity, gas, water, and telephony services) and private and public customers.

The Utility Statement supplements an Invoice with information about consumption of the utility's services. An invoice may refer to one or more utility statements, and a utility statement may refer to one or more invoices.

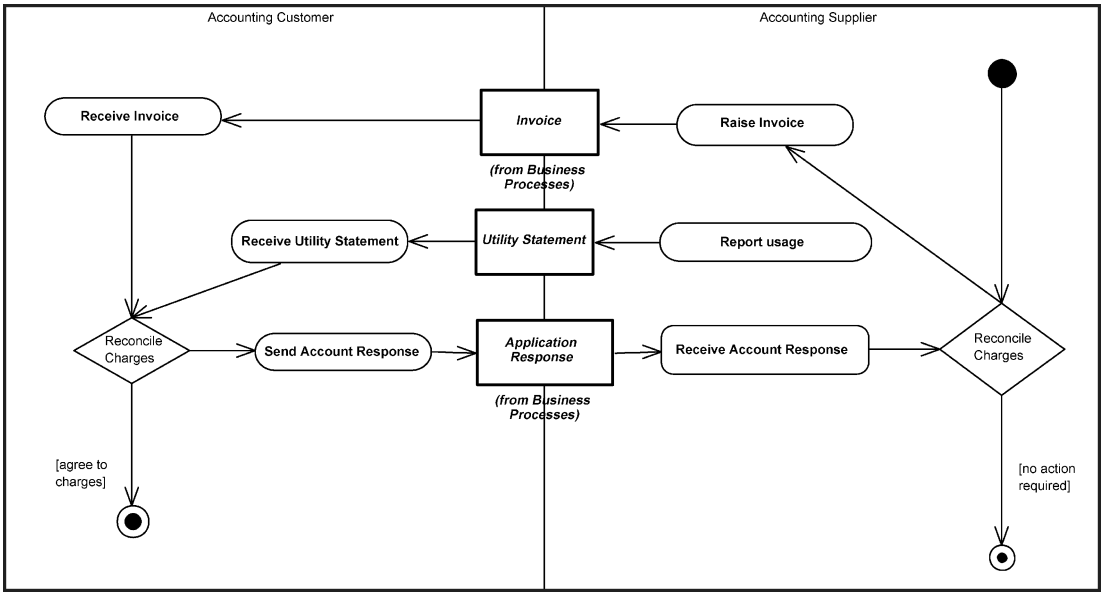

Figure 72 — Utility Billing Process

5.3.7.4 Payment Notification

In the payment notification process, the Payee (who is most often the Accounting Customer) is notified of any funds transferred, against the account of the Accounting Supplier, using a Remittance Advice document.

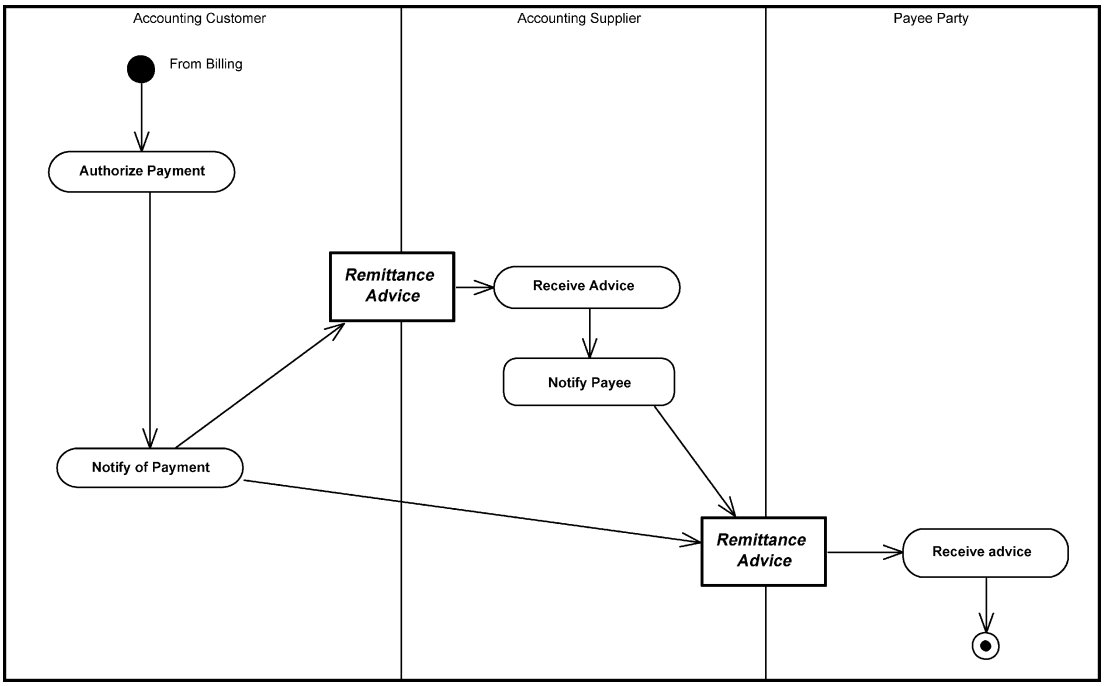

Figure 73 — Payment Notification Process

5.3.7.5 Report State of Accounts

A Statement of account may be used to notify the Accounting Customer of the status of the billing.

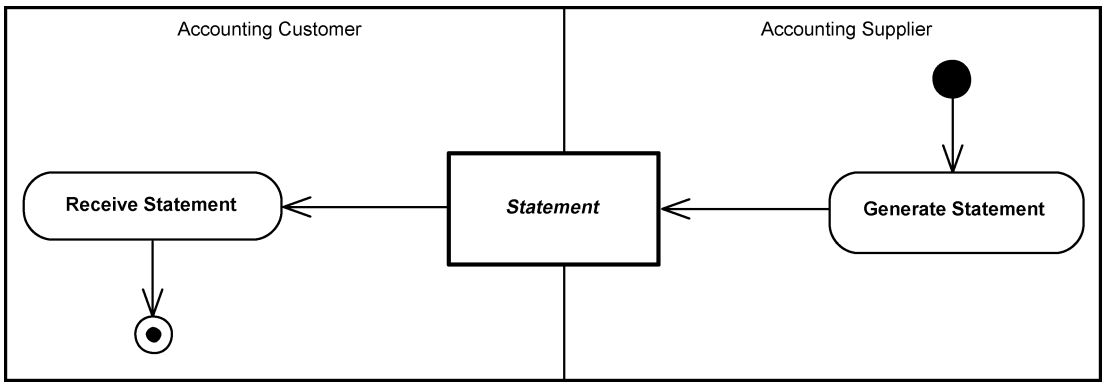

Figure 74 — Statement Process

5.3.8 Business Directory and Agreements

5.3.8.1 Directory Introduction

One of the increasing challenges with undertaking digital business is discovering and recording the specific operational and technical capabilities of trading organizations to reciprocate in digital trading agreements that are interoperable. As the market relies less and less on single service provider hubs and moves to a federated 4-corner model for document exchanges, this information becomes distributed across various parties.

5.3.8.2 Business Card

The Business Card allows a standardized way of presenting general trading capability information as well as company's main communication channels and references to company presentations such as flyers and brochures.

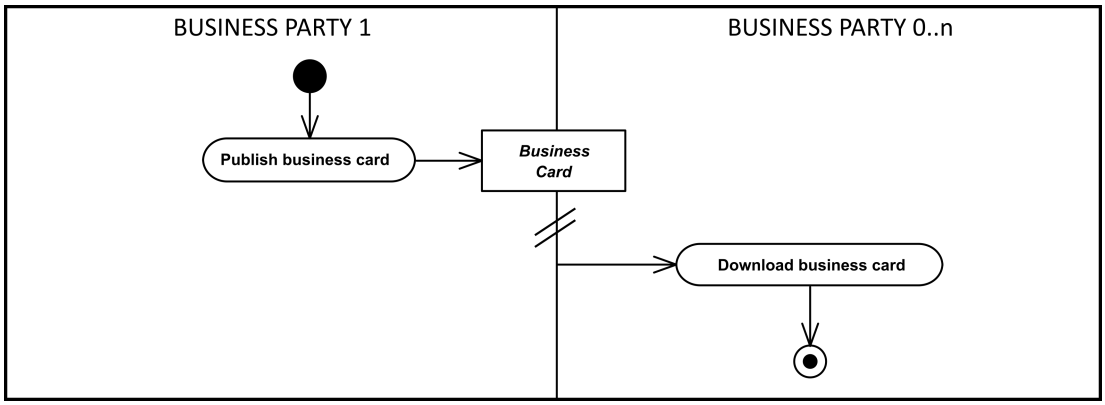

Figure 75 — Business card process

5.3.8.3 Digital Capability

The Digital Capability allows a standardized way of presenting digital trading capability ratification in a form that can be published or exchanged with trading partners. The digital capabilities of business partners are the source for building a Digital Agreement.

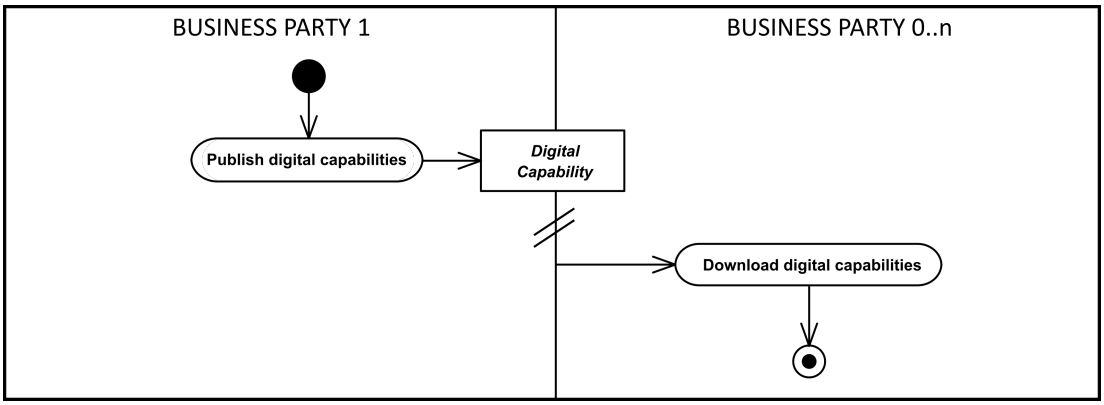

Figure 76 — Digital capability process

The data structures have been derived from the work of ebXML CPPA (Collaboration Protocol Profile and Agreement), OpenPEPPOL and other directory services initiatives.

5.3.8.4 Digital Agreement

Bi-lateral and multi-lateral trading partner agreements can make use of the standardized Digital Agreement document used to support business parties agreeing on a set of digital processes, terms and conditions.

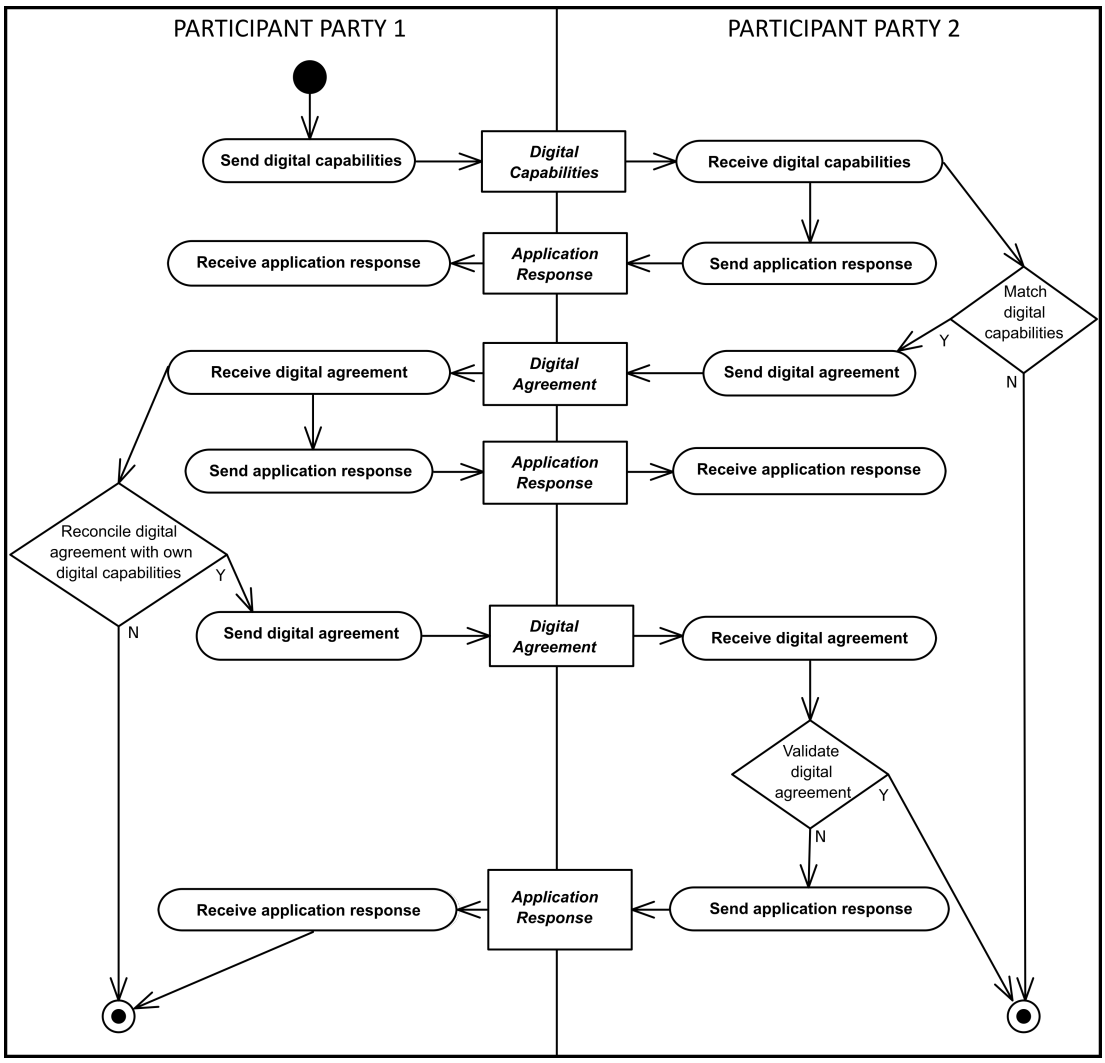

Figure 77 — Bi-lateral TPA process

5.4 Party Roles

In the UBL supply chain processes, two main actors, Customer and Supplier, represent the key organizations or people involved in the processes. Each of these actors may play various roles. Some processes may also involve supplementary roles that may be provided by different parties.

The actual role undertaken is dependent on the context of use. For example, the Despatch Party and Delivery Party as applied to the Procurement process may differ in the Transportation process. In the Transportation Process, two of the main roles are the Transport User and the Transport Service Provider. The Transport User is the role responsible for purchasing a transport service, while the Transport Service Provider is the role responsible for selling and executing a transport service. Both the Customer and the Supplier may be responsible for purchasing and following up the transport of goods, hence both these actors may undertake the Transport User role. In other words, the role of a specific actor depends on the specific circumstances.

The following table contains a description of the typical roles for the actor known as Party. Note that some roles require an extension of the information entities required. In UBL, the following are roles that extend the Party structure: Customer Party, Supplier Party, Contracting Party, Endorser Party, and Qualifying Party.

Actor	Role	Description	Example	Synonyms	Sends	Receives
Customer Party	Originator	The party that had the original demand for the goods and/or services and therefore initiated the procurement transaction. The Originator participates in pre-ordering activity either through Request for Quotation and Quotation or by receiving a Quotation as a response to a punch-out transaction on a marketplace or Seller's website. If the Originator subsequently places an Order, the Originator adopts the role of Buyer. The Originator is typically the contact point for queries regarding the original requirement and may be referred to in an Order Change, Order Cancellation, or Order Response.	If an employee requests a computer, the employing company may become the Buyer, but the employee is the Originator. They need to receive information about the order.		Request for Quotation	Quotation
Customer Party	Buyer	The party that purchases the goods or services on behalf of the Originator. The Buyer may be referred to in Order Response, Despatch Advice, Fulfilment Cancellation, Invoice, Self Billed Invoice, Credit Note, and Statement.	A company may delegate the task of purchasing to a specialized group to consolidate orders and gain greater discounts.	Order Point	Order, Order Change, Order Cancellation, Fulfilment Cancellation	Order Response, Fulfilment Cancellation
Customer Party	Delivery	The party to whom goods should be delivered. The Delivery Party may be the same as the Originator. The Delivery Party must be referred to at line item level in Request for Quotation, Quotation, Order, Order Change, Order Cancellation, and Order Response. The Delivery Party may be referred to at line level in Invoice, Self Billed Invoice, Credit Note, and Debit Note. The Delivery Party may be stipulated in a transport contract.	If a municipality buys a wheelchair for a citizen, the wheelchair must be delivered to the citizen (the Delivery Party). In such cases the citizen may be notified before delivery of the wheelchair.	Delivery Point, Destination Party, Receiver, Recipient	Receipt Advice	Despatch Advice
Customer Party	Accounting Customer	The party responsible for making settlement relating to a purchase and resolving billing issues using a Debit	If a kindergarten buys some toys they may be the Originator, Buyer,	Invoice, Accounts Payable, Debtor	In a traditional Billing scenario: Debit Note,	In a traditional Billing scenario:

(continued)

Actor	Role	Description	Example	Synonyms	Sends	Receives
		Note. The Accounting Customer must be referred to in an Order and may be referred to in an Order Response. In a Self Billing scenario, the Accounting Customer is responsible for calculating and issuing tax invoices.	and Delivery Party, but the municipality may play the role of Accounting Customer—they are going to pay for it.		Application Response, and Remittance Advice In a Self Billing scenario: Self Billed Invoice, Self Billed Credit Note, and Remittance Advice	Invoice, Credit Note, and Statement; in a Self Billing scenario: Credit Note, Application Response, and Statement
Supplier Party	Seller	The party responsible for handling Originator and Buyer services. The Seller party is legally responsible for providing the goods to the Buyer. The Seller party receives and quotes against Request for Quotation documents and may provide information to the Buyer's requisitioning process through Catalogues and Quotations.	The organization that sells wheelchairs to municipalities.	Sales Point, Provider, Customer Manager	Quotation, Order Response, Order Response Simple, Catalogue, Catalogue Deletion, Catalogue Item Specification Update, Catalogue Pricing Update, Fulfilment Cancellation	Request for Quotation, Order, Order Change, Order Cancellation, Catalogue Request, Fulfilment Cancellation
Supplier Party	Despatch	The party where goods are to be collected from. The Despatch Party may be stipulated in a transport contract.	The wheelchair Supplier may store chairs at a local warehouse. The warehouse will actually despatch the chair to the Delivery Party. The local warehouse is then the Despatch Party.	Despatch Point, Shipper, Sender	Despatch Advice	Receipt Advice
Supplier Party	Accounting Supplier	The party who claims the payment and is responsible for resolving billing issues and arranging settlement.	There are cases where the Accounting Supplier is not the Seller party. For example, factoring, where the invoicing is outsourced to another company.	Accounts Receivable, Invoice Issuer, Creditor	In a traditional Billing scenario: Invoice, Credit Note, and Statement; in a Self Billing scenario: Credit Note, Application Response, and Statement	In a traditional Billing scenario: Debit Note, Account Response, and Remittance Advice In a Self Billing scenario: Self Billed Invoice, Self Billed Credit Note, and Remittance Advice

(continued)

Actor	Role	Description	Example	Synonyms	Sends	Receives
Supplier Party	Payee	The party to whom the Invoice is paid.	The Accounting Supplier may not be the party to be paid due to changes in the organization, e.g., a company merger.	Accounts Receivable, Creditor		Remittance Advice
Customer Party	Contractor	The party responsible for the contract to which the Catalogue relates.	An organization has a central office for maintaining catalogues of approved items for purchase.	Central Catalogue Party, Purchasing Manager	Catalogue Request	Catalogue, Catalogue Deletion, Catalogue Item Specification Update, Catalogue Pricing Update
Party	Provider	The party responsible for the integrity of the information provided about an item.	The manufacturer may publish and maintain the data sheets about a product.		Catalogue, Catalogue Deletion, Catalogue Item Specification Update, Catalogue Pricing Update	
Party	Receiver	A general role, describing the receiver of a document. For a catalogue, this can be the customer, a potential customer, or a third party exposing the document, for instance, an interim broker.	A marketplace may receive an Application Response.			Catalogue, Catalogue Deletion, Catalogue Item Specification Update, Catalogue Pricing Update, Application Response
Party	Sender	The party sending a document.	A marketplace may send an Application Response.		Application Response	
Customer Party	Contracting Authority	The party responsible for making the contract relating to a tender ending up with a purchase.	If a kindergarten buys a lot of toys they may be a Contracting Authority in a Public Tender.	Customer, Debtor	Expression Of Interest Response, Qualification Application Request, Tender Contract, Tender Status, Unsubscribe From Procedure Response	Expression Of Interest Request, Qualification Application Response, and Tender Status Request, Tender Withdrawal, Unsubscribe From Procedure Request

(continued)

Actor	Role	Description	Example	Synonyms	Sends	Receives
Supplier Party	Tenderer	The party responsible for handling Originator and Buyer services. The Tenderer party is legally responsible for providing the goods to the Contracting Authority. The Tenderer party receives the Expression Of Interest Response.	The organization that sells wheelchairs to municipalities.	Seller, Provider, Economic Operator	Expression Of Interest Request, Qualification Application Response, Tender Status Request, Tender Withdrawal, Unsubscribe From Procedure Request	Expression Of Interest Response, Qualification Application Request, Tender Contract, Tender Status, Unsubscribe From Procedure Response
Party	Consignor	The party consigning the goods as stipulated in the transport contract. A Buyer, Delivery, Seller, or Despatcher Party may also play the role of Consignor. Also known as the Transport User. The Consignor may be stipulated in a transport contract.	The wheelchair Supplier may source from a local warehouse. The Freight Forwarder will collect the chair from the local warehouse, which is thus the Consignor. In this case, the warehouse also plays the role of Despatch Party to the Freight Forwarder.	Despatch Point, Shipper, Sender, Transport User	Forwarding Instructions, Packing List	Bill of Lading, Waybill, Freight Invoice, Transportation Status
Party	Consignee	The party receiving a consignment of goods as stipulated in the transport contract.	The party taking responsibility for the receipt of the consignment covering the wheelchair.	Delivery Point, Transport Service Buyer	Forwarding Instructions, Freight Invoice	Bill of Lading, Waybill, Freight Invoice, Transportation Status
Party	Freight Forwarder	The party arranging the carriage of goods, including connected services and/or associated formalities, on behalf of a Consignor or Consignee. Also known as the Transport Service Provider. The Freight Forwarder may also be the Carrier. The Freight Forwarder may create an Invoice and bill to the Transport Service Buyer for the transportation service provided.	The Consignor may have a contract with this Freight Forwarder, which is a Transport Services Provider, to arrange all their transport needs.	Shipping Agent, Broker, Courier, Transport Service Provider	Forwarding Instructions, Freight Invoice, Transportation Status	Bill of Lading, Waybill, Packing List
Party	Carrier	The party providing physical transport services.	The Freight Forwarder may engage an airline company to deliver the wheelchair. The airline is then the Carrier and delivers	Freight Haulier, Shipper, Ships Agent, Shipping Company, Airline, Rail Operator,	Bill of Lading, Waybill	Forwarding Instructions

(continued)

Actor	Role	Description	Example	Synonyms	Sends	Receives
			the chair to the Delivery Party.	Road Haulier		
Party	Exporter	The party who makes regulatory export declarations, or on whose behalf regulatory export declarations are made, and who is the owner of the goods or has similar right of disposal over them at the time when the declaration is accepted.	The wheelchair Supplier has to apply for a Certificate of Origin in order to sell the chairs overseas.	Seller, Consignor	Certificate of Origin	Application Response
Party	Endorser	The party appointed by the Government of a country who has the right to certify a Certificate of Origin. This endorsement restricts goods imported from certain countries for political or other reasons.	The Government agency validates all the information provided by Exporter for Certificate of Origin approval.	Authorized Organization, Embassy	Certificate of Origin, Application Response	Certificate of Origin
Party	Importer	The party who makes, or on whose behalf an agent or other authorized person makes, an import declaration. This may include a person who has possession of the goods or to whom the goods are consigned.	A specialized group in a company consolidates the purchase request and handles the receiving of goods.	Order Point, Delivery Party, Buyer, Customer, Consignee		Certificate of Origin
Party	Transport User	The Transport User is the role representing anyone who has a demand for transport services, books transport services, and follows up the execution of such services.	The manufacturer has to order transport of products from a carrier or freight forwarder (Transport Service Provider).	Transport Buyer, Logistics Service Client	Transport Execution Plan Request, Transportation Status Request, Transport Service Description Request	Transport Execution Plan, Transportation Status, Transport Service Description, Goods Item Itinerary
Party	Transport Service Provider	The Transport Service Provider is the role that plans, markets and performs transport services.	The carrier or freight forwarder who arranges for transport services on behalf of a manufacturer (Transport User)	Transport Provider, Transport Seller, Logistics Service Provider	Transport Execution Plan, Transportation Status, Transport Service Description, Transport Progress Status Request, Goods Item Itinerary	Transport Execution Plan Request, Transportation Status Request, Transport Service Description Request, Transport Progress Status
Party	Transportation Network Manager	The Transportation Network Manager is the role that extracts all information available regarding the infrastructure	The Traffic Information Centre (TIC) issuing information related to road work and/or	Road Administration, Traffic Information Centre,	Transport Progress Status	Transport Progress Status Request

(continued)

Actor	Role	Description	Example	Synonyms	Sends	Receives
		(static/dynamic) related to planning and executing transport and makes this information available to the Transport Service Provider. During a transport service, or even during a single leg, the Transport Service Provider may rely on information from several Transportation Network Managers.	traffic conditions as a service to a Transport Service Provider	Coastal Administration, Harbor Master, Railway Administration, Infrastructure Manager		
Party	Governor	The Governor is the role that governs an agreement or contract.	A legal entity who creates and maintain an agreement.			
Party	Participant	The Participant is the role agreeing on a set of digital processes, terms and conditions to ensure interoperability within a business network. A Buyer, Seller, Accounting Customer, Accounting Supplier, Service Provider Party may also play the role of Participant. A Participant in the role of a Business Party communicates its digital capabilities using a Digital Capability document.	A Service Provider agreeing on multi-lateral trading partner agreement governed by an e-Procurement network.		Digital Agreement, Application Response	Digital Agreement, Application Response
Party	Business	The Business Party is a general role that may be played by any other Party doing business according to a set of business and digital capabilities. A Business Party communicates its business information and capabilities to other parties using a Business Card. A Business Party communicates its digital capabilities to other parties using a Digital Capability document.	A Business Party supports the procurement business process according to a specific profile governed by an UBL user group.	Trading Partner, Service Provider, Economic Operator, Contracting Authority, Participant	Business Card, Digital Capability, Application Response	Business Card, Digital Capability, Application Response
Party	Weighing	The Weighing Party is a role played by weighing stations, shippers, terminal operators and possibly other parties executing a weight measurement including verified gross mass measurements.	A Business Party supports the procurement business process according to a specific profile governed by an UBL user group.	Weighing Station, Weighing Provider	Weight Statement	Application Response
Party	Responsible	The party responsible for signing the VGM on behalf of the Shipper.	A Weighing Party playing the role of a			

(continued)

Actor	Role	Description	Example	Synonyms	Sends	Receives
			Responsible who signs a VGM.			

6 UBL 2.2 Schemas

6.1 UBL 2.2 Schemas Introduction

The UBL XSD schemas [5][6] are the only normative representations of the UBL document types and library components for the purposes of XML [3] document validation and conformance.

All of the UBL XSD schemas are contained in the `xsd` subdirectory of the UBL release package (see Annex A Release Notes for more information regarding the structure of the release package and 6.4 Schema Dependencies for information regarding dependencies among the schema modules). The `xsd` directory is further subdivided into an `xsd/maindoc` subdirectory containing the schemas for individual document types and an `xsd/common` subdirectory containing schemas in the UBL common library. For convenience in implementing the schemas, parallel (and technically non-normative) "runtime" sets with the annotation elements stripped out are provided in the `xsdrt/` directory.

6.2 UBL 2.2 Document Schemas

6.2.1 UBL 2.2 Document Schemas Introduction

The tables that follow describe each of the UBL document types. Along with a link to the normative schema for each document type, each table provides links to the corresponding "runtime" schema, model spreadsheets and summary report in HTML (see Annex C The UBL 2.2 Data Model), and example instance, if any (see Annex F UBL 2.2 Example Document Instances).

6.2.2 Application Response Schema

Description: A document to indicate the application's response to a transaction. This may be a business response and/or a technical response, sent automatically by an application or initiated by a user.

Processes involved	Any collaboration
Submitter role	Sender
Receiver role	Receiver
Normative schema	`xsd/maindoc/UBL-ApplicationResponse-2.2.xsd`
Runtime schema	`xsdrt/maindoc/UBL-ApplicationResponse-2.2.xsd`
Summary report	`mod/summary/reports/UBL-ApplicationResponse-2.2.html`

6.2.3 Attached Document Schema

Description: A UBL wrapper that allows a document of any kind to be packaged with the UBL document that references it.

Processes involved	Any collaboration
Submitter role	Sender
Receiver role	Receiver
Normative schema	`xsd/maindoc/UBL-AttachedDocument-2.2.xsd`
Runtime schema	`xsdrt/maindoc/UBL-AttachedDocument-2.2.xsd`
Summary report	`mod/summary/reports/UBL-AttachedDocument-2.2.html`

6.2.4 Awarded Notification Schema

Description: The document used to communicate a contract award to the winner.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer

Normative schema	xsd/maindoc/UBL-AwardedNotification-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-AwardedNotification-2.2.xsd
Summary report	mod/summary/reports/UBL-AwardedNotification-2.2.html

6.2.5 Bill Of Lading Schema

Description: A document that conveys information about an instance of a transportation service and may under some circumstances serve as a contractual document For the service. See Bill of Lading and compare with Waybill.

Processes involved	Transport
Submitter role	Freight Forwarder, Carrier
Receiver role	Consignor (or Consignee), Freight Forwarder
Normative schema	xsd/maindoc/UBL-BillOfLading-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-BillOfLading-2.2.xsd
Summary report	mod/summary/reports/UBL-BillOfLading-2.2.html

6.2.6 Business Card Schema

Description: A document used to provide information about a business party and its business capabilities.

Processes involved	Business Directory and Agreements
Submitter role	Sender
Receiver role	Receiver
Normative schema	xsd/maindoc/UBL-BusinessCard-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-BusinessCard-2.2.xsd
Summary report	mod/summary/reports/UBL-BusinessCard-2.2.html
UBL 2.2 example instance	xml/UBL-BusinessCard-2.2-Example.xml

6.2.7 Call For Tenders Schema

Description: A document used by a Contracting Party to define a procurement project to buy goods, services, or works during a specified period.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer
Normative schema	xsd/maindoc/UBL-CallForTenders-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-CallForTenders-2.2.xsd
Summary report	mod/summary/reports/UBL-CallForTenders-2.2.html

6.2.8 Catalogue Schema

Description: A document that describes items, prices, and price validity. See Catalogue.

Processes involved	Catalogue, Create Catalogue, Delete Catalogue, Update Catalogue Item Specification, Update Catalogue Pricing, Initial Stocking of the Area by Producer, Permanent Replenishment, Price Adjustments, Transfer of Base Item Catalogue, Changes to the Item Catalogue, Changes to the Article Catalogue
Submitter role	Seller
Receiver role	Contracting Party
Normative schema	xsd/maindoc/UBL-Catalogue-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Catalogue-2.2.xsd
Summary report	mod/summary/reports/UBL-Catalogue-2.2.html

6.2.9 Catalogue Deletion Schema

Description: A document used to cancel an entire Catalogue.

Processes involved	Catalogue
Submitter role	Seller
Receiver role	Contracting Party
Normative schema	xsd/maindoc/UBL-CatalogueDeletion-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-CatalogueDeletion-2.2.xsd
Summary report	mod/summary/reports/UBL-CatalogueDeletion-2.2.html

6.2.10 Catalogue Item Specification Update Schema

Description: A document used to update information (e.g., technical descriptions and properties) about Items in an existing Catalogue.

Processes involved	Catalogue
Submitter role	Seller
Receiver role	Contracting Party
Normative schema	xsd/maindoc/UBL-CatalogueItemSpecificationUpdate-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-CatalogueItemSpecificationUpdate-2.2.xsd
Summary report	mod/summary/reports/UBL-CatalogueItemSpecificationUpdate-2.2.html

6.2.11 Catalogue Pricing Update Schema

Description: A document used to update information about prices in an existing Catalogue.

Processes involved	Catalogue
Submitter role	Seller
Receiver role	Contracting Party
Normative schema	xsd/maindoc/UBL-CataloguePricingUpdate-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-CataloguePricingUpdate-2.2.xsd
Summary report	mod/summary/reports/UBL-CataloguePricingUpdate-2.2.html

6.2.12 Catalogue Request Schema

Description: A document used to request a Catalogue.

Processes involved	Catalogue
Submitter role	Contracting Party
Receiver role	Seller
Normative schema	xsd/maindoc/UBL-CatalogueRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-CatalogueRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-CatalogueRequest-2.2.html

6.2.13 Certificate Of Origin Schema

Description: A document that describes the Certificate of Origin.

Processes involved	Certification of Origin of Goods
Submitter role	Exporter, Issuer
Receiver role	Issuer, Importer
Normative schema	xsd/maindoc/UBL-CertificateOfOrigin-2.2.xsd

Runtime schema	xsdrt/maindoc/UBL-CertificateOfOrigin-2.2.xsd
Summary report	mod/summary/reports/UBL-CertificateOfOrigin-2.2.html

6.2.14 Contract Award Notice Schema

Description: A document published by a Contracting Party to announce the awarding of a procurement project.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer
Normative schema	xsd/maindoc/UBL-ContractAwardNotice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ContractAwardNotice-2.2.xsd
Summary report	mod/summary/reports/UBL-ContractAwardNotice-2.2.html

6.2.15 Contract Notice Schema

Description: A document used by a Contracting Party to announce a project to buy goods, services or works.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer
Normative schema	xsd/maindoc/UBL-ContractNotice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ContractNotice-2.2.xsd
Summary report	mod/summary/reports/UBL-ContractNotice-2.2.html

6.2.16 Credit Note Schema

Description: A document used to specify credits due to the Debtor from the Creditor.

Processes involved	Billing
Submitter role	Supplier Accounting Party
Receiver role	Customer Accounting Party
Normative schema	xsd/maindoc/UBL-CreditNote-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-CreditNote-2.2.xsd
Summary report	mod/summary/reports/UBL-CreditNote-2.2.html
UBL 2.0 example instance	xml/UBL-CreditNote-2.0-Example.xml
UBL 2.1 example instance	xml/UBL-CreditNote-2.1-Example.xml

6.2.17 Debit Note Schema

Description: A document used to specify debts incurred by the Debtor.

Processes involved	Billing
Submitter role	Customer Accounting Party
Receiver role	Supplier Accounting Party
Normative schema	xsd/maindoc/UBL-DebitNote-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-DebitNote-2.2.xsd
Summary report	mod/summary/reports/UBL-DebitNote-2.2.html
UBL 2.1 example instance	xml/UBL-DebitNote-2.1-Example.xml

6.2.18 Despatch Advice Schema

Description: A document used to describe the despatch or delivery of goods and services.

Processes involved	Logistics
Submitter role	Despatch
Receiver role	Delivery
Normative schema	xsd/maindoc/UBL-DespatchAdvice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-DespatchAdvice-2.2.xsd
Summary report	mod/summary/reports/UBL-DespatchAdvice-2.2.html
UBL 2.0 example instance	xml/UBL-DespatchAdvice-2.0-Example.xml

6.2.19 Digital Agreement Schema

Description: A document used to support business parties agreeing on a set of digital processes, terms and conditions to ensure interoperability.

Processes involved	Business Directory and Agreements
Submitter role	Agreement Participant
Receiver role	Agreement Participant
Normative schema	xsd/maindoc/UBL-DigitalAgreement-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-DigitalAgreement-2.2.xsd
Summary report	mod/summary/reports/UBL-DigitalAgreement-2.2.html
UBL 2.2 example instance	xml/UBL-DigitalAgreement-2.2-Example.xml
UBL 2.2 example instance	xml/UBL-DigitalAgreement-2.2-Example-Multilateral.xml

6.2.20 Digital Capability Schema

Description: A document used to provide information about a business party and its digital capabilities.

Processes involved	Business Directory and Agreements
Submitter role	Sender
Receiver role	Receiver
Normative schema	xsd/maindoc/UBL-DigitalCapability-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-DigitalCapability-2.2.xsd
Summary report	mod/summary/reports/UBL-DigitalCapability-2.2.html
UBL 2.2 example instance	xml/UBL-DigitalCapability-2.2-Example.xml

6.2.21 Document Status Schema

Description: A document used to provide information about document status.

Processes involved	Any collaboration
Submitter role	Party currently controlling Status of the collaboration
Receiver role	Party requesting Status on collaboration
Normative schema	xsd/maindoc/UBL-DocumentStatus-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-DocumentStatus-2.2.xsd
Summary report	mod/summary/reports/UBL-DocumentStatus-2.2.html

6.2.22 Document Status Request Schema

Description: A document used to request the status of another document.

Processes involved	Any collaboration
Submitter role	Party requesting Status on collaboration
Receiver role	Party currently controlling Status of the collaboration
Normative schema	xsd/maindoc/UBL-DocumentStatusRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-DocumentStatusRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-DocumentStatusRequest-2.2.html

6.2.23 Enquiry Schema

Description: A document sent by a requestor to a responder requesting information about a particular business process.

Processes involved	Any collaboration
Submitter role	Requestor
Receiver role	Responder
Normative schema	xsd/maindoc/UBL-Enquiry-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Enquiry-2.2.xsd
Summary report	mod/summary/reports/UBL-Enquiry-2.2.html

6.2.24 Enquiry Response Schema

Description: A document sent by a responder to a requester answering a particular enquiry.

Processes involved	Any collaboration
Submitter role	Responder
Receiver role	Requestor
Normative schema	xsd/maindoc/UBL-EnquiryResponse-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-EnquiryResponse-2.2.xsd
Summary report	mod/summary/reports/UBL-EnquiryResponse-2.2.html

6.2.25 Exception Criteria Schema

Description: A document used to specify the thresholds for forecast variance, product activity, and performance history beyond which exceptions should be triggered.

Processes involved	Collaborative Planning, Forecasting, and Replenishment
Submitter role	Buyer, Seller
Receiver role	Buyer, Seller
Normative schema	xsd/maindoc/UBL-ExceptionCriteria-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ExceptionCriteria-2.2.xsd
Summary report	mod/summary/reports/UBL-ExceptionCriteria-2.2.html
UBL 2.1 example instance	xml/UBL-ExceptionCriteria-2.1-Example.xml

6.2.26 Exception Notification Schema

Description: A document used to notify an exception in forecast variance, product activity, or performance history.

Processes involved	Collaborative Planning, Forecasting, and Replenishment
Submitter role	Buyer, Seller
Receiver role	Buyer, Seller
Normative schema	xsd/maindoc/UBL-ExceptionNotification-2.2.xsd

Runtime schema	xsdrt/maindoc/UBL-ExceptionNotification-2.2.xsd
Summary report	mod/summary/reports/UBL-ExceptionNotification-2.2.html
UBL 2.1 example instance	xml/UBL-ExceptionNotification-2.1-Example.xml

6.2.27 Expression Of Interest Request Schema

Description: A document whereby an Economic Operator (the tenderer) makes an Expression Of Interest in a Call For Tenders to a Contracting Authority

Processes involved	Tendering (pre-award)
Submitter role	Tenderer (Economic Operator)
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-ExpressionOfInterestRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ExpressionOfInterestRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-ExpressionOfInterestRequest-2.2.html
UBL 2.2 example instance	xml/UBL-ExpressionOfInterestRequest-2.2-Example.xml

6.2.28 Expression Of Interest Response Schema

Description: A document whereby a Contracting Authority accepts receiving an Expression Of Interest from an Economic Operator (the tenderer)

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer (Economic Operator)
Normative schema	xsd/maindoc/UBL-ExpressionOfInterestResponse-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ExpressionOfInterestResponse-2.2.xsd
Summary report	mod/summary/reports/UBL-ExpressionOfInterestResponse-2.2.html

6.2.29 Forecast Schema

Description: A document used to forecast sales or orders.

Processes involved	Collaborative Planning, Forecasting, and Replenishment
Submitter role	Buyer, Seller
Receiver role	Buyer, Seller
Normative schema	xsd/maindoc/UBL-Forecast-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Forecast-2.2.xsd
Summary report	mod/summary/reports/UBL-Forecast-2.2.html
UBL 2.1 example instance	xml/UBL-Forecast-2.1-Example.xml

6.2.30 Forecast Revision Schema

Description: A document used to revise a Forecast.

Processes involved	Collaborative Planning, Forecasting, and Replenishment
Submitter role	Buyer, Seller
Receiver role	Buyer, Seller
Normative schema	xsd/maindoc/UBL-ForecastRevision-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ForecastRevision-2.2.xsd
Summary report	mod/summary/reports/UBL-ForecastRevision-2.2.html
UBL 2.1 example instance	xml/UBL-ForecastRevision-2.1-Example.xml

6.2.31 Forwarding Instructions Schema

Description: A document issued to a forwarder, giving instructions regarding the action to be taken for the forwarding of goods described therein. See Forwarding Instructions.

Processes involved	Transport
Submitter role	Consignor (or Consignee), Freight Forwarder
Receiver role	Freight Forwarder, Carrier
Normative schema	xsd/maindoc/UBL-ForwardingInstructions-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ForwardingInstructions-2.2.xsd
Summary report	mod/summary/reports/UBL-ForwardingInstructions-2.2.html
UBL 2.0 example instance	xml/UBL-ForwardingInstructions-2.0-Example-International.xml

6.2.32 Freight Invoice Schema

Description: A document stating the charges incurred for a logistics service.

Processes involved	Freight Billing
Submitter role	Freight Forwarder
Receiver role	Consignor or Consignee
Normative schema	xsd/maindoc/UBL-FreightInvoice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-FreightInvoice-2.2.xsd
Summary report	mod/summary/reports/UBL-FreightInvoice-2.2.html
UBL 2.1 example instance	xml/UBL-FreightInvoice-2.1-Example.xml

6.2.33 Fulfilment Cancellation Schema

Description: A document used to cancel an entire Despatch Advice or Receipt Advice.

Processes involved	Logistics
Submitter role	Buyer or Seller
Receiver role	Seller or Buyer
Normative schema	xsd/maindoc/UBL-FulfilmentCancellation-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-FulfilmentCancellation-2.2.xsd
Summary report	mod/summary/reports/UBL-FulfilmentCancellation-2.2.html
UBL 2.1 example instance	xml/UBL-FulfilmentCancellation-2.1-Example.xml

6.2.34 Goods Item Itinerary Schema

Description: A document providing details relating to a transport service, such as transport movement, identification of equipment and goods, subcontracted service providers, etc.

Processes involved	Intermodal Freight Management
Submitter role	Transport Service Provider
Receiver role	Transport User
Normative schema	xsd/maindoc/UBL-GoodsItemItinerary-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-GoodsItemItinerary-2.2.xsd
Summary report	mod/summary/reports/UBL-GoodsItemItinerary-2.2.html
UBL 2.1 example instance	xml/UBL-GoodsItemItinerary-2.1-Example.xml

6.2.35 Guarantee Certificate Schema

Description: A document to notify the deposit of a bid bond guarantee.

Processes involved	Tendering (pre-award)
Submitter role	Tenderer
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-GuaranteeCertificate-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-GuaranteeCertificate-2.2.xsd
Summary report	mod/summary/reports/UBL-GuaranteeCertificate-2.2.html

6.2.36 Instruction For Returns Schema

Description: A document used to initiate a return of goods. The producer is requesting the return of products that are not selling well, either to use in other places or to free up rack or shelf space.

Processes involved	Cyclic Replenishment Program (CRP)
Submitter role	Seller
Receiver role	Buyer
Normative schema	xsd/maindoc/UBL-InstructionForReturns-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-InstructionForReturns-2.2.xsd
Summary report	mod/summary/reports/UBL-InstructionForReturns-2.2.html
UBL 2.1 example instance	xml/UBL-InstructionForReturns-2.1-Example.xml

6.2.37 Inventory Report Schema

Description: A report on the quantities of each item that are, or will be, in stock. This document is sent by a Buyer (for example a retailer) to a Seller (for example a producer).

Processes involved	Cyclic Replenishment Program (CRP)
Submitter role	Buyer
Receiver role	Seller
Normative schema	xsd/maindoc/UBL-InventoryReport-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-InventoryReport-2.2.xsd
Summary report	mod/summary/reports/UBL-InventoryReport-2.2.html
UBL 2.1 example instance	xml/UBL-InventoryReport-2.1-Example.xml

6.2.38 Invoice Schema

Description: A document used to request payment.

Processes involved	Billing
Submitter role	Supplier Accounting Party
Receiver role	Customer Accounting Party
Normative schema	xsd/maindoc/UBL-Invoice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Invoice-2.2.xsd
Summary report	mod/summary/reports/UBL-Invoice-2.2.html
UBL 2.0 example instance	xml/UBL-Invoice-2.0-Example.xml
UBL 2.1 example instance	xml/UBL-Invoice-2.1-Example.xml
UBL 2.1 example instance	xml/UBL-Invoice-2.1-Example-Trivial.xml

6.2.39 Item Information Request Schema

Description: A document used to request product activity, forecast, or performance data.

Processes involved	Collaborative Planning, Forecasting, and Replenishment
Submitter role	Buyer, Seller
Receiver role	Buyer, Seller
Normative schema	xsd/maindoc/UBL-ItemInformationRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ItemInformationRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-ItemInformationRequest-2.2.html

6.2.40 Order Schema

Description: A document used to order goods and services.

Processes involved	Ordering (post-award)
Submitter role	Buyer
Receiver role	Seller
Normative schema	xsd/maindoc/UBL-Order-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Order-2.2.xsd
Summary report	mod/summary/reports/UBL-Order-2.2.html
UBL 2.0 example instance	xml/UBL-Order-2.0-Example.xml
UBL 2.1 example instance	xml/UBL-Order-2.1-Example.xml
UBL 2.0 example instance	xml/UBL-Order-2.0-Example-International.xml

6.2.41 Order Cancellation Schema

Description: A document used to cancel an entire Order.

Processes involved	Ordering (post-award), Logistics
Submitter role	Buyer
Receiver role	Seller
Normative schema	xsd/maindoc/UBL-OrderCancellation-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-OrderCancellation-2.2.xsd
Summary report	mod/summary/reports/UBL-OrderCancellation-2.2.html
UBL 2.1 example instance	xml/UBL-OrderCancellation-2.1-Example.xml

6.2.42 Order Change Schema

Description: A document used to specify changes to an existing Order.

Processes involved	Ordering (post-award), Logistics
Submitter role	Buyer
Receiver role	Seller
Normative schema	xsd/maindoc/UBL-OrderChange-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-OrderChange-2.2.xsd
Summary report	mod/summary/reports/UBL-OrderChange-2.2.html
UBL 2.1 example instance	xml/UBL-OrderChange-2.1-Example.xml

6.2.43 Order Response Schema

Description: A document used to indicate detailed acceptance or rejection of an Order or to make a counter-offer.

Processes involved	Ordering (post-award)
Submitter role	Seller
Receiver role	Buyer
Normative schema	xsd/maindoc/UBL-OrderResponse-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-OrderResponse-2.2.xsd
Summary report	mod/summary/reports/UBL-OrderResponse-2.2.html
UBL 2.1 example instance	xml/UBL-OrderResponse-2.1-Example.xml

6.2.44 Order Response Simple Schema

Description: A document used to indicate simple acceptance or rejection of an entire Order.

Processes involved	Ordering (post-award)
Submitter role	Seller
Receiver role	Buyer
Normative schema	xsd/maindoc/UBL-OrderResponseSimple-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-OrderResponseSimple-2.2.xsd
Summary report	mod/summary/reports/UBL-OrderResponseSimple-2.2.html
UBL 2.0 example instance	xml/UBL-OrderResponseSimple-2.0-Example.xml
UBL 2.1 example instance	xml/UBL-OrderResponseSimple-2.1-Example.xml

6.2.45 Packing List Schema

Description: A document describing how goods are packed.

Processes involved	Transport
Submitter role	Consignor
Receiver role	Freight Forwarder
Normative schema	xsd/maindoc/UBL-PackingList-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-PackingList-2.2.xsd
Summary report	mod/summary/reports/UBL-PackingList-2.2.html

6.2.46 Prior Information Notice Schema

Description: A document used by a contracting party to declare the intention to buy goods, services, or works during a specified period.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer
Normative schema	xsd/maindoc/UBL-PriorInformationNotice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-PriorInformationNotice-2.2.xsd
Summary report	mod/summary/reports/UBL-PriorInformationNotice-2.2.html
UBL 2.2 example instance	xml/UBL-PriorInformationNotice-2.2-Example-Embedded.xml
UBL 2.2 example instance	xml/UBL-PriorInformationNotice-2.2-Example-External.xml

6.2.47 Product Activity Schema

Description: A document reporting the movement of goods at specified retail locations for inventory tracking purposes.

Processes involved	Collaborative Planning, Forecasting, and Replenishment, Vendor Managed Inventory
Submitter role	Buyer
Receiver role	Seller
Normative schema	xsd/maindoc/UBL-ProductActivity-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ProductActivity-2.2.xsd
Summary report	mod/summary/reports/UBL-ProductActivity-2.2.html
UBL 2.1 example instance 1	xml/UBL-ProductActivity-2.1-Example-1.xml
UBL 2.1 example instance 2	xml/UBL-ProductActivity-2.1-Example-2.xml
UBL 2.1 example instance 3	xml/UBL-ProductActivity-2.1-Example-3.xml

6.2.48 Qualification Application Request Schema

Description: A document whereby a Contracting Authority makes a description of the required qualification Application (In Europe: ESPD Request) to an Economic Operator (the tenderer)

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer (Economic Operator)
Normative schema	xsd/maindoc/UBL-QualificationApplicationRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-QualificationApplicationRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-QualificationApplicationRequest-2.2.html

6.2.49 Qualification Application Response Schema

Description: A document whereby an Economic Operator (the tenderer) makes a description of the required qualification Application (In Europe: ESPD Response) to an Contracting Authority

Processes involved	Tendering (pre-award)
Submitter role	Tenderer (Economic Operator)
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-QualificationApplicationResponse-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-QualificationApplicationResponse-2.2.xsd
Summary report	mod/summary/reports/UBL-QualificationApplicationResponse-2.2.html

6.2.50 Quotation Schema

Description: A document used to quote for the provision of goods and services.

Processes involved	Quotation
Submitter role	Seller
Receiver role	Originator
Normative schema	xsd/maindoc/UBL-Quotation-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Quotation-2.2.xsd
Summary report	mod/summary/reports/UBL-Quotation-2.2.html
UBL 2.0 example instance	xml/UBL-Quotation-2.0-Example.xml
UBL 2.1 example instance	xml/UBL-Quotation-2.1-Example.xml

6.2.51 Receipt Advice Schema

Description: A document used to describe the receipt of goods and services.

Processes involved	Logistics
Submitter role	Delivery
Receiver role	Despatch
Normative schema	xsd/maindoc/UBL-ReceiptAdvice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-ReceiptAdvice-2.2.xsd
Summary report	mod/summary/reports/UBL-ReceiptAdvice-2.2.html
UBL 2.0 example instance	xml/UBL-ReceiptAdvice-2.0-Example.xml

6.2.52 Reminder Schema

Description: A document used to remind a customer of payments overdue.

Processes involved	Billing
Submitter role	Supplier Accounting Party and/or Payee
Receiver role	Customer Accounting Party and/or Payee
Normative schema	xsd/maindoc/UBL-Reminder-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Reminder-2.2.xsd
Summary report	mod/summary/reports/UBL-Reminder-2.2.html
UBL 2.1 example instance	xml/UBL-Reminder-2.1-Example.xml

6.2.53 Remittance Advice Schema

Description: A document that specifies details of an actual payment.

Processes involved	Payment Notification
Submitter role	Supplier Accounting Party and/or Payee
Receiver role	Customer Accounting Party and/or Payee
Normative schema	xsd/maindoc/UBL-RemittanceAdvice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-RemittanceAdvice-2.2.xsd
Summary report	mod/summary/reports/UBL-RemittanceAdvice-2.2.html
UBL 2.0 example instance	xml/UBL-RemittanceAdvice-2.0-Example.xml

6.2.54 Request For Quotation Schema

Description: A document used to request a Quotation for goods and services from a seller.

Processes involved	Quotation
Submitter role	Originator
Receiver role	Seller
Normative schema	xsd/maindoc/UBL-RequestForQuotation-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-RequestForQuotation-2.2.xsd
Summary report	mod/summary/reports/UBL-RequestForQuotation-2.2.html
UBL 2.0 example instance	xml/UBL-RequestForQuotation-2.0-Example.xml
UBL 2.1 example instance	xml/UBL-RequestForQuotation-2.1-Example.xml

6.2.55 Retail Event Schema

Description: A document used to specify basic information about retail events (such as promotions, product introductions, and community or environmental events) that affect supply or demand.

Processes involved	Cyclic Replenishment Program (CRP)
Submitter role	Buyer, Seller
Receiver role	Buyer, Seller
Normative schema	xsd/maindoc/UBL-RetailEvent-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-RetailEvent-2.2.xsd
Summary report	mod/summary/reports/UBL-RetailEvent-2.2.html
UBL 2.1 example instance	xml/UBL-RetailEvent-2.1-Example.xml

6.2.56 Self Billed Credit Note Schema

Description: A credit note created by the debtor in a self billing arrangement with a creditor; Self Billed Credit Note replaces Debit Note in such arrangements.

Processes involved	Billing
Submitter role	Customer Accounting Party
Receiver role	Supplier Accounting Party
Normative schema	xsd/maindoc/UBL-SelfBilledCreditNote-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-SelfBilledCreditNote-2.2.xsd
Summary report	mod/summary/reports/UBL-SelfBilledCreditNote-2.2.html
UBL 2.1 example instance	xml/UBL-SelfBilledCreditNote-2.1-Example.xml

6.2.57 Self Billed Invoice Schema

Description: An invoice document created by the customer (rather than the supplier) in a Self Billing relationship.

Processes involved	Billing
Submitter role	Customer Accounting Party
Receiver role	Supplier Accounting Party
Normative schema	xsd/maindoc/UBL-SelfBilledInvoice-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-SelfBilledInvoice-2.2.xsd
Summary report	mod/summary/reports/UBL-SelfBilledInvoice-2.2.html

6.2.58 Statement Schema

Description: A document used to report the status of orders, billing, and payment. This document is a statement of account, not a summary invoice.

Processes involved	Billing
Submitter role	Supplier Accounting Party
Receiver role	Customer Accounting Party
Normative schema	xsd/maindoc/UBL-Statement-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Statement-2.2.xsd
Summary report	mod/summary/reports/UBL-Statement-2.2.html
UBL 2.0 example instance	xml/UBL-Statement-2.0-Example.xml

6.2.59 Stock Availability Report Schema

Description: A report on the quantities of each item that are, or will be, in stock. This document is sent by a Seller (for example a producer) to a Buyer (for example a retailer).

Processes involved	Cyclic Replenishment Program (CRP)
Submitter role	Seller (Producer)
Receiver role	Buyer (Retailer)
Normative schema	xsd/maindoc/UBL-StockAvailabilityReport-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-StockAvailabilityReport-2.2.xsd
Summary report	mod/summary/reports/UBL-StockAvailabilityReport-2.2.html
UBL 2.1 example instance	xml/UBL-StockAvailabilityReport-2.1-Example.xml

6.2.60 Tender Schema

Description: A document whereby an economic operator (the tenderer) makes a formal offer (the tender) to a contracting authority to execute an order for the supply or purchase of goods, or for the execution of work, according to the terms of a proposed contract.

Processes involved	Tendering (pre-award)
Submitter role	Tenderer
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-Tender-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Tender-2.2.xsd
Summary report	mod/summary/reports/UBL-Tender-2.2.html

6.2.61 Tender Contract Schema

Description: A document whereby a Contracting Authority sends information to the Economic Operator describing the final contract after a tendering procedure.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer (Economic Operator)
Normative schema	xsd/maindoc/UBL-TenderContract-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TenderContract-2.2.xsd
Summary report	mod/summary/reports/UBL-TenderContract-2.2.html

6.2.62 Tender Receipt Schema

Description: A document sent by a contracting party to an economic operator acknowledging receipt of a Tender.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer
Normative schema	xsd/maindoc/UBL-TenderReceipt-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TenderReceipt-2.2.xsd
Summary report	mod/summary/reports/UBL-TenderReceipt-2.2.html

6.2.63 Tender Status Schema

Description: A document whereby a Contracting Authority sends information to the Economic Operator describing the status of a tendering procedure.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer (Economic Operator)
Normative schema	xsd/maindoc/UBL-TenderStatus-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TenderStatus-2.2.xsd
Summary report	mod/summary/reports/UBL-TenderStatus-2.2.html

6.2.64 Tender Status Request Schema

Description: A document whereby an Economic Operator (the tenderer) asking about the details and status of a tendering procedure

Processes involved	Tendering (pre-award)
Submitter role	Tenderer (Economic Operator)
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-TenderStatusRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TenderStatusRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-TenderStatusRequest-2.2.html

6.2.65 Tender Withdrawal Schema

Description: A document whereby an Economic Operator (the tenderer) makes a Tender Withdrawal to a Contracting Authority

Processes involved	Tendering (pre-award)
Submitter role	Tenderer (Economic Operator)
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-TenderWithdrawal-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TenderWithdrawal-2.2.xsd
Summary report	mod/summary/reports/UBL-TenderWithdrawal-2.2.html

6.2.66 Tenderer Qualification Schema

Description: A document declaring the qualifications of a tenderer.

Processes involved	Tendering (pre-award)
Submitter role	Tenderer
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-TendererQualification-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TendererQualification-2.2.xsd
Summary report	mod/summary/reports/UBL-TendererQualification-2.2.html

6.2.67 Tenderer Qualification Response Schema

Description: A document issued by a procurement organization to notify an economic operator whether it has been admitted to or excluded from the tendering process.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer

Normative schema	xsd/maindoc/UBL-TendererQualificationResponse-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TendererQualificationResponse-2.2.xsd
Summary report	mod/summary/reports/UBL-TendererQualificationResponse-2.2.html

6.2.68 Trade Item Location Profile Schema

Description: A document specifying trade item attributes relating to replenishment policies.

Processes involved	Collaborative Planning, Forecasting, and Replenishment
Submitter role	Buyer, Seller
Receiver role	Buyer, Seller
Normative schema	xsd/maindoc/UBL-TradeItemLocationProfile-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TradeItemLocationProfile-2.2.xsd
Summary report	mod/summary/reports/UBL-TradeItemLocationProfile-2.2.html
UBL 2.1 example instance	xml/UBL-TradeItemLocationProfile-2.1-Example.xml

6.2.69 Transport Execution Plan Schema

Description: A document used in the negotiation of a transport service between a transport user and a transport service provider.

Processes involved	Intermodal Freight Management
Submitter role	Transport Service Provider, Transport User
Receiver role	Transport User, Transport Service Provider
Normative schema	xsd/maindoc/UBL-TransportExecutionPlan-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportExecutionPlan-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportExecutionPlan-2.2.html
UBL 2.1 example instance	xml/UBL-TransportExecutionPlan-2.1-Example.xml

6.2.70 Transport Execution Plan Request Schema

Description: A document sent by a transport user to request a transport service from a transport service provider.

Processes involved	Intermodal Freight Management
Submitter role	Transport User
Receiver role	Transport Service Provider
Normative schema	xsd/maindoc/UBL-TransportExecutionPlanRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportExecutionPlanRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportExecutionPlanRequest-2.2.html
UBL 2.1 example instance	xml/UBL-TransportExecutionPlanRequest-2.1-Example.xml

6.2.71 Transport Progress Status Schema

Description: A document sent from a transportation network manager to a transport service provider giving the status of the whereabouts and schedule of the transport means involved in a transport service.

Processes involved	Intermodal Freight Management
Submitter role	Transportation Network Manager
Receiver role	Transport Service Provider
Normative schema	xsd/maindoc/UBL-TransportProgressStatus-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportProgressStatus-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportProgressStatus-2.2.html
UBL 2.1 example instance	xml/UBL-TransportProgressStatus-2.1-Example.xml

6.2.72 Transport Progress Status Request Schema

Description: A document sent from a transport service provider to a transportation network manager requesting a Transport Progress Status.

Processes involved	Intermodal Freight Management
Submitter role	Transport Service Provider
Receiver role	Transportation Network Manager
Normative schema	xsd/maindoc/UBL-TransportProgressStatusRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportProgressStatusRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportProgressStatusRequest-2.2.html
UBL 2.1 example instance	xml/UBL-TransportProgressStatusRequest-2.1-Example.xml

6.2.73 Transport Service Description Schema

Description: A document sent by a transport service provider to announce the availability of a transport service.

Processes involved	Intermodal Freight Management
Submitter role	Transport Service Provider
Receiver role	Transport User
Normative schema	xsd/maindoc/UBL-TransportServiceDescription-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportServiceDescription-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportServiceDescription-2.2.html
UBL 2.1 example instance	xml/UBL-TransportServiceDescription-2.1-Example.xml

6.2.74 Transport Service Description Request Schema

Description: A document requesting a Transport Service Description, sent from a party with a transport demand (transport user) to a party providing transport services (transport service provider).

Processes involved	Intermodal Freight Management
Submitter role	Transport User
Receiver role	Transport Service Provider
Normative schema	xsd/maindoc/UBL-TransportServiceDescriptionRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportServiceDescriptionRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportServiceDescriptionRequest-2.2.html
UBL 2.1 example instance	xml/UBL-TransportServiceDescriptionRequest-2.1-Example.xml

6.2.75 Transportation Status Schema

Description: A document to circulate reports of transportation status or changes in status (events) among a group of participants.

Processes involved	Freight Status Reporting
Submitter role	Transport Service Provider
Receiver role	Transport User
Normative schema	xsd/maindoc/UBL-TransportationStatus-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportationStatus-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportationStatus-2.2.html
UBL 2.1 example instance	xml/UBL-TransportationStatus-2.1-Example.xml

6.2.76 Transportation Status Request Schema

Description: A document requesting a Transportation Status report.

Processes involved	Freight Status Reporting
Submitter role	Transport User
Receiver role	Transport Service Provider
Normative schema	xsd/maindoc/UBL-TransportationStatusRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-TransportationStatusRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-TransportationStatusRequest-2.2.html
UBL 2.1 example instance	xml/UBL-TransportationStatusRequest-2.1-Example.xml

6.2.77 Unawarded Notification Schema

Description: A document communicating to a tenderer that the contract has been awarded to different tenderer.

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer
Normative schema	xsd/maindoc/UBL-UnawardedNotification-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-UnawardedNotification-2.2.xsd
Summary report	mod/summary/reports/UBL-UnawardedNotification-2.2.html

6.2.78 Unsubscribe From Procedure Request Schema

Description: A document whereby an Economic Operator (the tenderer) wants to Unsubscribe From Procedure and sends it to Contracting Authority

Processes involved	Tendering (pre-award)
Submitter role	Tenderer (Economic Operator)
Receiver role	Contracting Authority
Normative schema	xsd/maindoc/UBL-UnsubscribeFromProcedureRequest-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-UnsubscribeFromProcedureRequest-2.2.xsd
Summary report	mod/summary/reports/UBL-UnsubscribeFromProcedureRequest-2.2.html

6.2.79 Unsubscribe From Procedure Response Schema

Description: A document whereby a Contracting Authority accepts receiving an Unsubscribe From Procedure from an Economic Operator (the tenderer) and sends a confirmation

Processes involved	Tendering (pre-award)
Submitter role	Contracting Authority
Receiver role	Tenderer (Economic Operator)
Normative schema	xsd/maindoc/UBL-UnsubscribeFromProcedureResponse-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-UnsubscribeFromProcedureResponse-2.2.xsd
Summary report	mod/summary/reports/UBL-UnsubscribeFromProcedureResponse-2.2.html

6.2.80 Utility Statement Schema

Description: A supplement to an Invoice or Credit Note, containing information on the consumption of services provided by utility suppliers to private and public customers, including electricity, gas, water, and telephone services.

Processes involved	Utility Billing
Submitter role	Supplier Accounting Party
Receiver role	Customer Accounting Party
Normative schema	xsd/maindoc/UBL-UtilityStatement-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-UtilityStatement-2.2.xsd
Summary report	mod/summary/reports/UBL-UtilityStatement-2.2.html

6.2.81 Waybill Schema

Description: A transport document describing a shipment. It is issued by the party who undertakes to provide transportation services, or undertakes to arrange for their provision, to the party who gives instructions for the transportation services (shipper, consignor, etc.). It states the instructions for the beneficiary and may contain the details of the transportation, charges, and terms and conditions under which the transportation service is provided. See Waybill and compare with Bill of Lading.

Processes involved	Transport
Submitter role	Freight Forwarder, Carrier
Receiver role	Consignor (or Consignee), Freight Forwarder
Normative schema	xsd/maindoc/UBL-Waybill-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-Waybill-2.2.xsd
Summary report	mod/summary/reports/UBL-Waybill-2.2.html
UBL 2.0 example instance	xml/UBL-Waybill-2.0-Example-International.xml

6.2.82 Weight Statement Schema

Description: A document used to report weight or verified mass measurements in the transport chain.

Processes involved	Transport
Submitter role	Sender
Receiver role	Receiver
Normative schema	xsd/maindoc/UBL-WeightStatement-2.2.xsd
Runtime schema	xsdrt/maindoc/UBL-WeightStatement-2.2.xsd
Summary report	mod/summary/reports/UBL-WeightStatement-2.2.html
UBL 2.2 example instance	xml/UBL-WeightStatement-2.2-Example.xml

6.3 UBL 2.2 Common Schemas

6.3.1 UBL 2.2 Common Schemas Introduction

The `xsd/common` directory contains schemas referenced by the document schemas in `xsd/maindoc`. Elements defined in the common schemas constitute a library of reusable business data components from which the UBL document schemas are (and customized document types may be) assembled. For a discussion of the way schemas are assembled, see Annex C The UBL 2.2 Data Model.

The name of each schema file together with a brief description of its contents is given below.

6.3.2 Reusable BIE Schemas

CommonBasicComponents

xsd/common/UBL-CommonBasicComponents-2.2.xsd

The CommonBasicComponents schema defines the global Basic Business Information Entities (BBIEs) that are used throughout UBL, serving, in effect, as a "global BBIE type database" for constructing documents. BBIEs are the "leaf nodes" of UBL documents, corresponding to individual data fields in traditional printed business forms.

CommonAggregateComponents

xsd/common/UBL-CommonAggregateComponents-2.2.xsd

The CommonAggregateComponents schema defines the Aggregate Business Information Entities (ABIEs) that are used throughout UBL, serving, in effect, as an "ABIE type database" for constructing the main documents.

For a discussion of the terms Basic Business Information Entity and Aggregate Business Information Entity, see C.4 Business Information Entities.

6.3.3 Reusable Data Type Schemas

CCTS_CCT_SchemaModule

xsd/common/CCTS_CCT_SchemaModule-2.2.xsd

This schema provides Core Component Types as defined by [12]. These types are used to construct higher-level data types in a standardized and consistent manner. This schema is defined by UN/CEFACT and should not be modified. It is imported by the UBL Unqualified Data Type Schema, and its types are the basis upon which UBL's unqualified data types are defined.

UnqualifiedDataTypes

xsd/common/UBL-UnqualifiedDataTypes-2.2.xsd

This schema defines Unqualified Data Types for BBIE definition. These types are derived from the Core Component Types in `CCTS_CCT_SchemaModule`. Where an unqualified type is not based solely on an XSD data type, all CCTS supplementary components are made available in the UBL UDT from the CCTS CCT.

QualifiedDataTypes

xsd/common/UBL-QualifiedDataTypes-2.2.xsd

[12] permits the definition of Qualified Datatypes as derivations from CCTS-specified Unqualified Datatypes. In UBL 2.2, all data type qualifications are expressed in the [16] file cva/UBL-DefaultDTQ-2.2.cva. The UBL-QualifiedDataTypes-2.2.xsd file in the UBL 2.2 release has declarations for each qualified type being only an unmodified restriction of the base unqualified data type, thus adding no constraints. The Common Basic Components type declarations point to the XSD qualified types where the BBIEs are qualified in the CCTS model, but all BBIEs are effectively unqualified.

See Annex D Data Type Qualifications in UBL for information regarding UBL 2.2 data type derivation.

6.3.4 Extension Content Schemas

UBL extensions enable the validation of user-defined additions to the standard schemas, which are sometimes needed to satisfy legal requirements and can perform other useful functions as well. For further information regarding the UBL extension mechanism, see [15].

CommonExtensionComponents

xsd/common/UBL-CommonExtensionComponents-2.2.xsd

The CommonExtensionComponents schema defines the extension scaffolding used in all UBL document types, providing metadata regarding the use of an extension embedded in a UBL document instance (see 6.5 Extension Methodology and Validation).

ExtensionContentDatatype

xsd/common/UBL-ExtensionContentDataType-2.2.xsd

The ExtensionContentDataType schema specifies the actual structural constraints of the extension element containing the foreign non-UBL content. By default, the version of this schema provided in the UBL 2.2 distribution imports the UBL Signature Extension module and namespace (see 6.3.5 Signature Extension Schemas). This both enables support by default for advanced digital signatures and serves as an illustration of the way extensions are defined in UBL.

This is the only schema intended to be modified by a user when it is necessary to support the constraints of additional user-defined extension structures. This is accomplished by adding other schema import directives, as is already done for the signature extension and that extension's use of XAdES. Without adding additional directives, the user's constructs found under the extension point will not be validated.

No changes are required to the complex type declaration for `ExtensionContentType`. The original declaration is considered the normative declaration but may be modified by users to accommodate restrictions they impose on the presence of extensions. To promote interoperability, imposing such restrictions on the type declaration is not recommended.

6.3.5 Signature Extension Schemas

UBL 2.2 schemas are supplied with a predefined standard extension that supports advanced digital signatures; see 6.4 Schema Dependencies and 8.4 UBL Extension for Enveloped XML Digital Signatures for further information regarding the UBL extension supporting digital signatures such as XAdES.

CommonSignatureComponents

xsd/common/UBL-CommonSignatureComponents-2.2.xsd

The CommonSignatureComponents schema defines the scaffolding structures containing the IETF/W3C Digital Signature information XML elements related to either the entire document or particular signature business objects found within the document.

SignatureAggregateComponents

xsd/common/UBL-SignatureAggregateComponents-2.2.xsd

The SignatureAggregateComponents schema defines those Aggregate Business Information Entities (ABIEs) that are used for signature constructs not defined in the common library.

SignatureBasicComponents

xsd/common/UBL-SignatureBasicComponents-2.2.xsd

The SignatureBasicComponents schema defines those Basic Business Information Entities (BBIEs) that are used for signature constructs not defined in the common library.

For a discussion of the terms Basic Business Information Entity and Aggregate Business Information Entity, see C.4 Business Information Entities.

xmldsig-core-schema

xsd/common/UBL-xmldsig1-schema-2.2.xsd

This is a copy of the IETF/W3C Digital Signature core schema file, modified only to include a header and to import the renamed other digital signature schema fragments.

xmldsig-core-schema

xsd/common/UBL-xmldsig11-schema-2.2.xsd

This is a copy of the IETF/W3C Digital Signature core schema file, modified only to include a header.

xmldsig-core-schema

xsd/common/UBL-xmldsig-core-schema-2.2.xsd

This is a copy of the IETF/W3C Digital Signature core schema file, modified only to include a header and to remove the unnecessary PUBLIC and SYSTEM identifiers from the DOCTYPE.

XAdES01903v132-201601

xsd/common/UBL-XAdES01903v132-201601-2.2.xsd

This is a copy of the XAdES v1.3.2 schema file, modified only to change the importing URI for the XML digital signature core schema file.

The presence of this schema file does not oblige the use of XAdES. It is provided only as a convenience for those users who choose to include an XAdES extension inside of a digital signature.

XAdES01903v141-201601

xsd/common/UBL-XAdES01903v141-201601-2.2.xsd

This is a copy of the XAdES v1.4.1 schema file, modified only to change the importing URI for the XAdES v1.3.2 and the XML digital signature core schema files.

The presence of this schema file does not oblige the use of XAdES. It is provided only as a convenience for those users who choose to include an XAdES extension inside of a digital signature.

6.4 Schema Dependencies

The following diagram details the dependencies among the schema modules making up a UBL 2.2 document schema.

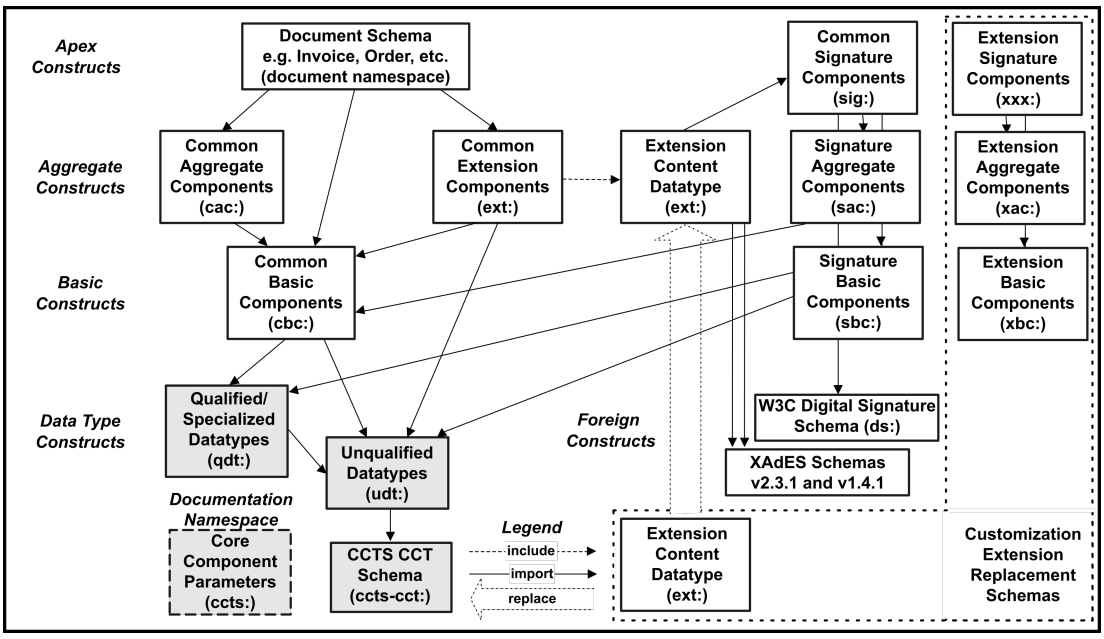

Figure 78 — UBL Schema Dependencies

The UBL schemas define in ExtensionContentDataType the content of each extension to be a single element in any namespace. The schemas are delivered supporting the UBL standardized extension for digital signatures (namespaces with prefixes `sig:`, `sac:` and `sbc:`, though the prefix values are not mandatory) by importation. For more information regarding the signature extension, see 8.4 UBL Extension for Enveloped XML Digital Signatures.

As shown at the bottom and right in this diagram, a set of XSD schemas supporting a different user-customized extension can be engaged by replacing the delivered ExtensionContentDataType schema fragment with one also importing the required custom schema apex fragment that defines the custom content (depicted using namespaces with example prefixes `xxx:`, `xac:` and `xbc:`).

The namespaces shown in the shaded boxes (with prefixes `qdt:`, `udt:`, `ccts-cct:` and `ccts:`) exist for the management of the schema components only and have no utility in UBL XML document instances. Declaring unused namespaces in an XML instance is superfluous and does not impact on conformance, but having them present may be confusing or misleading to the reader.

The relationship of the UBL schemas to the UBL data model is illustrated in Figure C.1.

6.5 Extension Methodology and Validation

6.5.1 Extension Methodology Overview

There exist many established XML vocabularies expressing useful semantics for information exchange. The W3C digital signature vocabulary is but one example of such a vocabulary that has its own governance, life-cycle and publication schedule. It is futile to attempt to mimic all of an established vocabulary's constructs as new UBL constructs and keep up with changes made in their life cycle. Moreover, it is untenable to ask users to re-frame all of the content of an established vocabulary into any such new UBL constructs.

Also, user communities may have the need to exchange information that is found neither in the UBL schemas nor in an established XML vocabulary. A colloquial XML vocabulary can be designed within which this information is expressed. Should the user community wish to promote the inclusion of their additional semantics into the UBL specification, the UBL Maintenance Governance Procedures [21] outlines how one would use the extension point and submit proposals for enhancements.

The UBL extension scaffolding allows the inclusion of multiple extensions in any UBL instance, be they structured by established or colloquial XML vocabularies.

6.5.2 Extension Expression

Every UBL instance is allowed to contain extension content using the element `<ext:UBLExtensions>` in the extension namespace `urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2` (there are no constraints on the namespace prefix, only the namespace URI). This element must be the first child element of the document element. It must contain one or more `<ext:UBLExtension>` elements.

Each `<ext:UBLExtension>` element contains the metadata and content of a single extension. All extension metadata is optional, and the extension content is mandatory. The extension content element contains as its only child the apex element, in a namespace other than the UBL extension namespace, of an arbitrary XML structure.

Element name	Card.	Type	Description
cbc:ID	0..1	Identifier	An identifier for the Extension assigned by the creator of the extension.
cbc:Name	0..1	Name	An identifier for the Extension assigned by the creator of the extension.
ext:ExtensionAgencyID	0..1	Identifier	An agency that maintains one or more Extensions.
ext:ExtensionAgencyName	0..1	Name	The name of the agency that maintains the Extension.
ext:ExtensionVersionID	0..1	Identifier	The version of the Extension.
ext:ExtensionAgencyURI	0..1	Identifier	A URI for the Agency that maintains the Extension.
ext:ExtensionURI	0..1	Identifier	A URI for the Extension.
ext:ExtensionReasonCode	0..1	Code	A code for reason the Extension is being included.
ext:ExtensionReason	0..1	Text	A description of the reason for the Extension.
ext:ExtensionContent	1	Element	The definition of the extension content.

An excerpt of the example instance xml/MyTransportationStatus.xml that includes a single extension without extension metadata is as follows:

```
<TransportationStatus
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns="urn:oasis:names:specification:ubl:schema:xsd:TransportationStatus-2">

  <!--this document needs additional information not defined by the UBL TC-->
  <ext:UBLExtensions>
    <ext:UBLExtension>
      <ext:ExtensionContent>
        <mec:Additional xmlns:mec="urn:X-MyCompany:Extension"
```

```
                        xmlns:mac="urn:X-MyCompany:Aggregate"
                        xmlns:mbc="urn:X-MyCompany:Basic">

          <mac:QualificationLevel>
            <cbc:ID>L1</cbc:ID>
            <cbc:Description>Level 1</cbc:Description>
            <mbc:LevelPrerequisite>Level0</mbc:LevelPrerequisite>
          </mac:QualificationLevel>
          ...
        </mec:Additional>
      </ext:ExtensionContent>
    </ext:UBLExtension>
  </ext:UBLExtensions>

  <!--the remainder is stock UBL-->
  <cbc:UBLVersionID>2.1</cbc:UBLVersionID>
  <cbc:CustomizationID>urn:X-demo:TransportShipments</cbc:CustomizationID>
  ...
</TransportationStatus>
```

6.5.3 Extension Validation

The UBL Digital Signature extension described in 8 UBL Digital Signatures is built into the UBL distribution and validates transparently.

Users wishing to validate other extensions found in the instance simply revise the `UBL-ExtensionContentDataType-2.2.xsd` schema fragment. An `<xsd:import>` directive is added to incorporate the schema constraints of the apex of another extension to be validated in the single pass of XSD validation. Figure 78 shows the replacement of the schema fragment with one in which user-defined extension modules with namespaces `ext:`, `xxx:`, `xac:`, and `xbc:` augment the digital signature extension modules with namespaces `ext:`, `sig:`, `sac:`, `sbc:` and `ds:`.

Due to limitations of W3C Schema validation semantics (this is not the case in RELAX NG [25], for example), the apex element of the extension in the instance being validated cannot be constrained solely to the apex element declared. W3C Schema's lax validation permits any element declared in any schema fragment to be the apex of an extension. Thus, an instance will pass when a known extension element not permitted by the user to be an apex element is in the place of an apex element. This is simply regarded by downstream processes as an unknown extension and will likely be ignored.

6.5.4 Notes For Extension Creators

The following points should be noted:

— Extension designers should follow the example by providing separate namespaces for apex element, aggregate constructs, and basic constructs if they wish the new items to be considered for inclusion in future UBL releases. This structures the new items for inclusion in the UBL common library. See `xml/MyTransportationStatus.xml` for a document instance exemplifying the recommended treatment of namespaces in a colloquial XML vocabulary.

— Whenever possible, one should use existing UBL common library aggregate and basic constructs in extensions rather than inventing new items with the same semantics. However, a common library aggregate construct should only be used when the entire aggregate and all of its descendants are applicable in the extension context without any changes. If any items must be removed, then a new extension aggregate with a new local name should be used. If all the constructs in the common library aggregate are applicable but some items need to be added, then a new extension aggregate with the same name should be created by adding the new constructs to a copy of the common library aggregate.

The UBL Digital Signature extension described in 8 UBL Digital Signatures has been modeled as an example to follow when designing and writing other custom extensions.

7 Additional Document Constraints

7.1 Additional Document Constraints Introduction

In addition to the UBL document constraints formally expressed by the schemas in 6 UBL 2.2 Schemas, UBL mandates several other rules governing conforming UBL instances that cannot be expressed using W3C Schema. These additional UBL document rules, addressing XML instance [3] validation, character encoding, and empty elements, are specified below.

NOTE These rules first appeared in the OASIS UBL 1.0 and UBL 1.0 NDR Standards. They are listed here because logically they belong with the great majority of UBL instance constraints specified in the schemas. To aid in coordinating references between these various publications, the rules below retain their original "IND" labels. The former IND4 was removed in the revision process leading to UBL 2.0.

Additional document constraints do not apply to the arbitrary content of extensions expressed in a UBL document as described in 6.5 Extension Methodology and Validation.

7.2 Validation

The UBL library and document schemas are targeted at supporting business information exchanges. Business information exchanges require a high degree of precision to ensure that application processing and corresponding business actions are reflective of the purpose, intent, and information content agreed to by both trading partners. Schemas provide the base mechanism for ensuring that instance documents do in fact support these requirements.

[IND1] All UBL instance documents MUST validate to a corresponding schema.

UBL recommends a two-phase approach for validation of rules related to specific data content (such as to check of code list values). See Annex E UBL 2.2 Code Lists and Two-phase Validation for a description of this approach.

7.3 Character Encoding

XML supports a wide variety of character encodings. Processors must understand which character encoding is employed in each XML document. XML 1.0 supports a default value of UTF-8 for character encoding, but best practice is always to identify the character encoding being employed.

[IND2] All UBL instance documents MUST identify their character encoding within the XML declaration.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

UBL, as an OASIS TC, is obligated to conform to agreements OASIS has entered into. OASIS is a liaison member of the ISO IEC ITU UN/CEFACT eBusiness Memorandum of Understanding Management Group (MOUMG). Resolution 01/08 (MOU/MG01n83) requires the use of UTF-8.

[IND3] In conformance with ISO IEC ITU UN/CEFACT eBusiness Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83) as agreed to by OASIS, all UBL XML SHOULD be expressed using UTF-8.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

7.4 Empty Elements

Use of empty elements within XML instance documents is a source of controversy for a variety of reasons. An empty element does not simply represent data that is missing. It may express data that is not applicable for some reason, trigger the expression of an attribute, denote all possible values instead of just one, mark the end of a series of data, or appear as a result of an error in XML file generation. Conversely, missing data elements can also have meaning—that the trading partner does not provide that data. In information exchange environments, different trading partners may allow, require, or ban empty elements. UBL takes the position that empty elements do not provide the level of assurance necessary for business information exchanges and therefore must not be used.

[IND5] UBL-conforming instance documents MUST NOT contain an element devoid of content or containing null values.

An important implication of this rule is that every container UBL element must contain at least one of its possible constituents even if all of its possible constituents are declared to be optional.

To ensure that no attempt is made to circumvent rule IND5, UBL also prohibits attempting to convey meaning by omitting an element (i.e., an optional element may be omitted, but that omission cannot carry a specific meaning upon which an action is conditioned).

[IND6] The absence of a construct or data in a UBL instance document MUST NOT carry meaning.

These constraints are consistent with the principle described in 5.2.2 Manifest Values that the recipient must receive all pertinent information manifest in the UBL document. Relying on the absence of a construct would require the recipient to know of the sender's intention with that construct being absent. For reliable communication this cannot be assumed.

7.5 Natural Language Text Elements

Natural language text elements such as Note and Description appear throughout the UBL document model. They are of the same unstructured Text type as character data fields that are not intended for natural language prose, such as Address Line.

All natural language text elements in UBL are repeatable within some container; for example, all Note elements are repeatable as adjacent siblings under a common parent. Despite appearances, these multiple text elements are not intended for the representation of separate paragraphs or divisions within a single parent text; rather, each Note element (for example) contains the entire text of the note in one of the languages in which the note is provided. In other words, UBL allows 0..n Note or Description elements in order to present the same note or description in 0..n languages, not to reflect structures such as paragraphs internal to a text in a single language. Since UBL text elements are intended for unstructured sequences of character data, more complex texts should be located in external documents and associated with the UBL message using document references.

UBL enforces this restriction with the following two rules:

[IND7] Where two or more sibling "Text. Type" elements of the same name exist in a document, no two can have the same "languageID" attribute value.

[IND8] Where two or more sibling "Text. Type" elements of the same name exist in a document, no two can omit the "languageID" attribute.

7.6 Empty Attributes

Attributes in UBL are used exclusively for supplemental components of the data types of basic business information entities. An empty attribute conveys no information but may be the source of confusion for users.

[IND7] UBL-conforming instance documents MUST NOT contain an attribute devoid of content or containing null values.

8 UBL Digital Signatures

8.1 UBL Digital Signatures Introduction

This section provides the context for the use of UBL digital signatures and then defines profiles for advanced digital signatures in UBL and a specific UBL extension that implements one specific kind of advanced digital signature.

There are certain circumstances in which it becomes necessary to electronically sign UBL documents. This can be the case when creating tenders or invoices. For example, in some countries digitally signing electronic invoices is required by law.

UBL (without extension) has a data structure (known as Signature) for defining electronic signatures and a number of elements for using such signatures in a document. To integrate UBL into the larger standards environment, this section associates the IETF/W3C XML Digital Signature specification [4] (a general framework for digitally signing XML documents) with the signature elements provided by UBL. These include specific provisions to use extensions supporting [2], XML Advanced Electronic Signatures (ETSI TS 101 903), when the electronic signing of UBL documents is necessary to satisfy legal and technical requirements.

XAdES extends XMLDSig for use with advanced and qualified electronic signatures as specified in European Directive [7]. Use of XAdES and the concept of Advanced Electronic Signature is not limited to Europe, as it is being adopted by many countries outside the EU, and, at the time of publication of this specification, it is undergoing international standardization in ISO as ISO 14533-2:2012 [30].

One important benefit of XAdES is that it allows the addition of information and timestamps that extend the validity of a signature beyond the expiration or revocation of the electronic certificates involved in signature verification or the obsolescence of the underlying cryptographic keys and algorithms. By extending XMLDSig with additional embedded syntax and processing, XAdES satisfies the European Directive on a Community Framework for Electronic Signatures as well as other use cases requiring long-term preservation of signed documents. XAdES contains several modules that permit various levels of security, such as content commitment and non-repudiation enforcement with timestamps and long-term signature verification.

The two digital signature profiles provided in UBL represent two approaches to signing UBL documents: enveloped and detached. Each of these approaches uses XMLDSig in a way that may or may not include XAdES features. In other words, the mechanisms implemented here can be used not only to implement XAdES in these two ways but also to implement other signature technologies based on XMLDSig as well.

8.2 XML Digital Signatures

8.2.1 XML Digital Signatures Overview

Digital signatures, when appropriate rules and functions are used, can support the following properties for a document:

— Integrity: the document has not been modified since it was signed.

— Authenticity: the identity of the party creating the signature that applies to the document is certified.

— Non-repudiation (content commitment): the document signer cannot deny its involvement in creating and/or approving the document (depending on the context and signer role).

— Anteriority: associating a time-stamp to the signature, a proof that the signature (and therefore the signed document) existed before a certain point in time.

[4] defines XML Signature processing rules and syntax to provide integrity and message authentication and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere. [26] specifies a standard format for time-stamping that can be used with XMLDSig and XAdES.

The [7] directive defines the following technology-neutral requirements that an electronic signature must meet to be considered an Advanced Electronic Signature (AdES) and have legal validity:

— it is uniquely linked to the signatory;

— it is capable of identifying the signatory;

— it is created using means that the signatory can maintain under his sole control; and

— it is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable.

The Qualified Signature (QS) is also defined as an AdES based on Qualified Certificates (QC) and Secure Signature Creation Devices for signing operations. In Europe, QS is equivalent to handwritten signature provided it is based on a QC issued by an accredited Certificate Service Provider. These references are provided only for informational use and refer to the framework defined in [7].

XAdES extends XMLDSig to support AdES, but its adoption is not limited to an EU context, as similar requirements are in place in other countries. The introduction to [2] reads, in part,

> The XML advanced electronic signatures defined in the present document will be built by incorporating to the XML signatures as defined in XMLDSIG one new `ds:Object` XML element containing the additional qualifying information.

That XAdES is completely embedded in XMLDSig ensures that the UBL profiles for XMLDSig are sufficient to support XAdES. These profiles also support other existing or future extensions of XMLDSig that are completely embedded in XMLDSig syntax. These other possible UBL digital signature profiles may or may not use the XAdES extensions to XMLDSig.

It is important to note that XAdES and XMLDSig define digital signature processing rules and syntax but do not cover the implementation of security measures required for an AdES, which are out of scope for UBL.

Implementation may depend on local regulations in place and specific provisions set by the authority issuing the certificates supporting the signature. The implementer has to determine the set of requirements that apply to the specific context of use and determine accordingly the suitability of the standards and the specific profiles to be used. XAdES can help in fulfilling legal requirements, but this is not just a matter of correctly applying a technical standard. Users are advised to examine the regulations applicable to their specific context of use.

8.2.2 XML Signature Types

An XML signature may be (non-exclusively) described (per XMLDSig and XAdES) as detached, enveloping, or enveloped.

— **Detached.** The signature applies to content that is external to the `<ds:Signature>` element and can be identified via a URI or transform. Consequently, the signature is "detached" from the content it signs. This definition typically applies to separate data objects, but it also includes the case where the `<ds:Signature>` and signed data object are sibling elements residing within the same XML document.

— **Enveloping.** The signature applies to content found within a `<ds:Object>` element of the signature itself. The `<ds:Object>` (or its content) is identified via a `<ds:Reference>` (using a URI fragment identifier or transform).

— **Enveloped.** The signature applies to the XML content that contains `<ds:Signature>` as an element. Implementations of enveloped signature(s) must take care not to include the signature in the calculation of the signature value.

UBL defines two profiles for signing a UBL document: enveloped and detached.

8.2.3 XAdES

A compliant implementation of XAdES guarantees wide acceptance in implementing legal regulations, such as European Commission Directive [7] and European Commission Decision [8], and it supports best practices in eInvoicing eProcurement eBusiness, eInvoicing eProcurement eBusiness, and eInvoicing eProcurement eBusiness in general as set forth by relevant standard bodies such as CEN ([18] and [17]).

The UBL implementation of XAdES provides the following additional properties:

— A signed UBL document will be processed correctly by any compliant UBL software (including UBL software that is not XMLDSig/XAdES aware) and by any compliant XMLDSig/XAdES verification software (including software that is not UBL aware).

— No change is required for currently defined UBL or XMLDSig/XAdES syntaxes.

— The extension mechanism specified here supports any XMLDSig/XAdES form, leaving to the implementer the choice of the most appropriate one according to the specific legal framework or application context.

XAdES defines a set of forms that extends XMLDSig and allows adding some validation data to the signature.

The two basic forms are:

— **XAdES-BES**, which satisfies the minimum requirements for AdES; and

— **XAdES-EPES**, which builds on XAdES-BES to include a security policy identifier that specifies the rules followed to validate the signature.

A conforming XAdES signature generation and verification implementation supports at least XAdES-BES or XAdES-EPES.

The other forms can be built by the signature generator or the signature verifier by extending one of the two basic forms. They are:

— **XAdES-T**, where a timestamp is added to enforce content commitment (non-repudiation) and as a proof of anteriority. This envelope allows ascertaining the validity of a signature in case the signer certificate is later revoked.

— **XAdES-C**, which adds to the signed document a complete reference to verification data (certificates and revocation lists) to support long-term signature verification.

— **XAdES-X**, which adds timestamps to XAdES-C references to protect against future compromise of certificates.

— **XAdES-X-L**, which is similar to XADES-X but adds real certificates and revocation lists instead of just references.

— **XAdES-A**, which adds timestamps (periodically, as required) to extend the validity period for long-term storage, taking into account a possible weakening of the algorithms used to sign the document and related certificates during the storage period.

No specific XAdES form is recommended for a UBL document, as this choice depends on the specific context of use, agreements between the parties, and local regulations.

8.2.4 Requirements for Digital Signatures in UBL

The main requirements to be addressed when choosing a specific signature profile can be divided into the following categories:

— **Legal requirements.** In some countries a digital signature is required on electronic invoices. It can also be compulsory in electronic procurement, especially in a cross-border context, to have a digital signature on the key document exchanged, e.g., a response to a request for tender. Another important legal requirement is long-term document preservation, for a storage period that in general is specific to each country and can span many years. The requirement to guarantee the integrity and authenticity of all fiscally relevant archived documents, as specified, for example, by [18] for electronic invoices, can be met with digital signatures when proper XAdES forms are used.

— **Business requirements.** A digital signature can reduce the risks associated with a business transaction (e.g., content commitment of a commercial order, proof-of-origin and integrity of an invoice), and its use can be provided for in the interchange agreement between parties. The choice of the signature format and its application is a key factor in achieving interoperability.

— **Process requirements.** The presence of the digital signature should not add any specific constraints on UBL document content processing. If the signed document remains a valid UBL document, the signature can be verified at any stage of the process: it should be possible to validate a signed document at any time “as is” by UBL and XAdES verifiers.

Archiving of UBL documents also can be an important issue to consider, as document preservation has specific requirements.

8.3 Profiles for UBL Digital Signatures

8.3.1 Signature Profile Introduction

UBL specifies two profiles for use in digitally signing UBL documents:

— **Enveloped Signature Profile:** One or more signatures are added to the UBL document inside a single identifiable and dedicated UBL extension. Other UBL extensions MAY be present provided they have different identifiers so that they can be distinguished from the one that contains the document signature(s). This profile is defined such that UBL content processing can be separated from electronic signature processing, both on the issuing side and on the receiving side, and specialized applications can be devoted to each function. The UBL application does not need to be electronic signature aware, and the electronic signature application does not need to be involved in the management of the UBL syntax. A signature business object in the UBL document may reference a particular electronic signature in the extension.

— **Detached Signature Profile:** The signature is outside the UBL document content in another information resource. Some mechanism has to be defined by the implementer to send or make available the signature to the recipient. This method of signing may be identified in the UBL document. This approach can be useful to avoid or minimize any kind of modification to the UBL document and is compatible with other signature methods not explicitly referenced by this profile.

The two profiles for adding one or more digital signatures to a UBL document are based on [4]. These profiles and their associated methods decouple the UBL document to be signed from any specificity in the digital signature standard adopted within XMLDSig. The XAdES standard is an example of a standard use of XMLDSig. UBL users may use any standard built on XMLDSig or simply use XMLDSig as it stands without any extensions.

Managing XML signatures inside of a UBL document is described in 8.3.2 Enveloped XML Signatures in UBL Documents. Managing XML signatures outside of a UBL document is described in 8.3.3 Detached XML Signatures for UBL Documents.

Both profiles support co-signatures, i.e., a UBL document can be independently cosigned by multiple signers in any order and at any time. Both profiles support countersignatures, i.e., a UBL document can have its signatures signed by another signature. The enveloped signature profile supports a final signature, i.e., a UBL document once signed with a final signature cannot have any other signature added without invalidating the final signature.

The choice of the most suitable profile should take into account the specific document processing and delivery infrastructure.

The main advantage of the enveloped profile is that the signature(s) are embedded in the UBL document (which syntactically remains a valid UBL document). This means that the transport of the signatures is guaranteed by the UBL document delivery infrastructure.

The detached signature profile has a simpler preparation phase and signature procedure, but specific means to send or make available the signature(s) to the recipient have to be implemented. A standard container like [24] can be used to associate the UBL document with detached advanced electronic signature(s) that apply to it. The simple [9] container (ASiC-S) can be created later than signature generation in such a way that it contains a UBL document and one or more detached signatures that apply to it.

8.3.2 Enveloped XML Signatures in UBL Documents

8.3.2.1 Enveloped Signature Introduction

The enveloped signature profile supports one or more signatures to be applied to a UBL document and embedded in the UBL document itself inside a dedicated extension. This profile can be used with all UBL documents under their respective `<ext:UBLExtensions>` extension points. UBL syntax implementing the enveloped profile, together with examples of its use, are provided in 8.4 UBL Extension for Enveloped XML Digital Signatures.

The user MAY choose to indicate in a `<cac:Signature>` element that the signature details are found in the signature extension. The URI `urn:oasis:names:specification:ubl:dsig:enveloped` is reserved as a value for `<cbc:SignatureMethod>` to signal this. The URI `urn:oasis:names:specification:ubl:dsig:enveloped:xades` MAY be used as a value for `<cbc:SignatureMethod>` to signal when XAdES is in use. Additionally, the user MAY include a `<cbc:ID>` child of `<cac:Signature>` for referencing purposes from the enveloped signature. The identifier used can be any value, but for convenience the URI of a URN beginning with `urn:oasis:names:specification:ubl:signature:`, ending with the local name of the parent of the signature business object, and optionally followed with a colon and number, as in the `urn:oasis:names:specification:ubl:signature:IssuerEndorsement` example, is reserved for this purpose for UBL users. As with all identifiers, the identifier SHOULD exist and SHOULD be unique across all identifier values. An example is as follows:

```
<cac:Signature>
  <cbc:ID>urn:oasis:names:specification:ubl:signature:Invoice</cbc:ID>
  <cbc:SignatureMethod
    >urn:oasis:names:specification:ubl:dsig:enveloped</cbc:SignatureMethod>
  <cac:SignatoryParty>
    <cac:PartyIdentification>
      <cbc:ID>MyParty</cbc:ID>
    </cac:PartyIdentification>
  </cac:SignatoryParty>
</cac:Signature>
```

See 8.5 Digital Signature Examples for a sample UBL Invoice that references an enveloped digital signature.

8.3.2.2 Enveloped Signature Syntax and Transformation

Two different syntaxes are used in UBL enveloped signatures: UBL-specified scaffolding under the extension point used to contain the signature information and IETF/W3C-specified information for each digital signature.

A transformation element is also present to prevent a signature from being invalidated by the subsequent addition of another signature.

These features are described in detail in 8.4.5 Digital Signature Structure and 8.4.6 Transformation.

8.3.3 Detached XML Signatures for UBL Documents

8.3.3.1 Detached Signature Introduction

This profile supports the application to a UBL document of one or more signatures located outside of the document itself in some other resource.

It is important to note that externally signing a UBL document with a detached signature imposes no requirements on the UBL document itself. Such a signature, in any kind of signature container, can digitally sign the content of a UBL document regardless of whether this is reflected in the document.

If a user knows the document will have a detached conforming IETF/W3C XML digital signature, the user MAY choose to signal in their UBL document that it is so signed. The URI value `urn:oasis:names:specification:ubl:dsig:detached` is reserved to indicate that the detached signature is an IETF/W3C XML digital signature. The URI `urn:oasis:names:specification:ubl:dsig:detached:xades` MAY be used as a value to signal when XAdES is in use. The value is used in the `<cbc:SignatureMethod>` child of `<cac:Signature>`.

If the location of the digital signature is known, the user MAY choose to indicate the location in a `<cbc:URI>` child element of a `<cac:ExternalReference>` child element of a `<cac:DigitalSignatureAttachment>` element.

Following is a complete example of a `<cac:Signature>` business object that might be found in a UBL instance:

```
<cac:Signature>
  <cbc:ID>urn:oasis:names:specification:ubl:signature:Invoice</cbc:ID>
  <cbc:SignatureMethod
    >urn:oasis:names:specification:ubl:dsig:detached</cbc:SignatureMethod>
  <cac:SignatoryParty>
    <cac:PartyIdentification>
      <cbc:ID>MyParty</cbc:ID>
    </cac:PartyIdentification>
  </cac:SignatoryParty>
  <cac:DigitalSignatureAttachment>
    <cac:ExternalReference>
      <cbc:URI>sigFile.xml</cbc:URI>
    </cac:ExternalReference>
  </cac:DigitalSignatureAttachment>
</cac:Signature>
```

NOTE A document with multiple detached signatures is simply a document that is co-signed. By the appropriate use of the `<ds:Reference>` element pointing to the UBL document from a detached signature file, all such signatures are signing the content of the document but not each other. A *countersigning* document signature, on the other hand, signs signatures already created for and external to or present in the document at the time it is countersigned. A digital countersignature `<ds:Signature>`, which may be located internal to the UBL document or in an external file, includes additional `<ds:Reference>` elements, each pointing either to the `<ds:Signature>` element or `<ds:SignatureValue>` element child of the signature being signed. In the first case, where the signature is detached, the `<ds:Reference>` element points to the external file for that signature; in the second case, where the signature is enveloped, the `<ds:Reference>` element points to the Id= value of either the `<ds:Signature>` or `<ds:SignatureValue>` element for that signature.

NOTE The XAdES specification supports an alternative countersignature approach where a `<ds:Signature>` element pointing to the countersigned signature's `<ds:SignatureValue>` is embedded in the `<ds:Object>` of the countersigning signature. The inclusion of an alternative method in this specification does not prohibit this approach.

See 8.5 Digital Signature Examples for a sample UBL Invoice that references a detached digital signature.

8.3.3.2 Digital Signature Transformation (Detached Signatures)

The content to be signed is addressed in the `URI=` attribute of `<ds:Reference>`:

```
<ds:Reference URI="myInvoice.xml">
```

An option when using detached digital signatures is to express in XPath that address that qualifies all nodes in the referenced content to be included in the calculation of the digital signature hash. For a signature calculated for a document to remain valid, none of the signed information can change, nor can any information be added or removed from that portion of the document included in the hash calculation.

Consider the need to create a detached signature for a UBL file in which there already exists an enveloped signature. The following transformation element in a digital signature flexibly prevents the signature being invalidated by the subsequent addition of any signatures using the enveloped profile within the extension of the document being signed:

```
  <Transform
    Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
   <XPath xmlns:sig=
```

```
      "urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2">
      count(ancestor-or-self::sig:UBLDocumentSignatures)=0
    </XPath>
  </Transform>
```

A non-final transformation algorithm used in the detached signature signs all content outside of any enveloped signatures in the UBL document. When the UBL document does not already have an enveloped signature, one cannot be added without invalidating the detached signature. In effect, the entire document has been signed and cannot change, but the addition of the scaffolding for a signature constitutes a change. However, when the UBL document already has an enveloped signature, other signatures can be added without invalidating the detached signature, because the scaffolding doesn't change when other signatures are added within the existing scaffolding; the non-final transformation algorithm does not include the signatures found in the existing scaffolding. When there is no preexisting enveloped signature, the entire document must be signed in the detached signature.

To sign only a portion of a UBL document, an appropriate [33] address SHOULD be used because UBL business object elements do not have attributes of type ID. This requires XPointer awareness on the part of the digital signature tools being used.

8.4 UBL Extension for Enveloped XML Digital Signatures

8.4.1 UBL Extension for Enveloped XML Digital Signatures Introduction

UBL extensions enable user-defined additions to the standard schemas. The UBL schemas in this distribution are provided with a predefined standard extension for enveloped signatures that supports IETF/W3C Digital Signature profiles. These include advanced IETF/W3C XML digital signatures conforming to the ETSI XAdES specification [2], thus satisfying EU legal requirements for electronically signed business documents.

This extension also serves as a case study for the creation of user-defined UBL extensions; see 6.5.4 Notes For Extension Creators. Further information on the UBL extension mechanism can be found in [15].

UBL's implementation of XML digital signatures puts all the signatures relating to a document in a single extension, which is engaged in validation by the `UBL-ExtensionContentDataType-2.2.xsd` schema module.

8.4.2 Digital Signature Namespaces

As is true for the UBL document schemas and common library, the UBL digital signature extension is modeled with three namespaces: one for the apex element (a parallel to the document schema), one for new aggregate constructs (a parallel to the common aggregate schema), and one for new basic constructs (a parallel to the common basic schema). See Figure 78.

The `urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2` namespace is used for the apex element, the `urn:oasis:names:specification:ubl:schema:xsd:SignatureAggregateComponents-2` namespace is used for new aggregate elements, and the `urn:oasis:names:specification:ubl:schema:xsd:SignatureBasicComponents-2` namespace is used for new basic elements. The IETF/W3C digital signature [4] standard namespace `http://www.w3.org/2000/09/xmldsig#` is also used in this extension. These namespaces are bound to the `sig:`, `sac:`, `sbc:` and `ds:` prefixes respectively, but any prefix or even the default namespace can be used for any of these in an XML instance.

Schema fragments for the two XAdES namespaces `http://uri.etsi.org/01903/v1.3.2#` and `http://uri.etsi.org/01903/v1.4.1#` are included in UBL for the convenience of users of the XAdES specification. There is no obligation to use the XAdES extension in the IETF/W3C digital signature. The appropriate XSD fragments are imported into the overall schema structure from the extension content data type schema fragment. Changing UBL to support a future version of the XAdES schema fragments involves only changing the import statements in the extension content data type schema fragment.

The table below lists the namespaces used for UBL digital signatures. The prefixes on the left are only documentary conventions; their choice is not constrained by XML.

Prefix	Namespace	Reference
`ds`	`http://www.w3.org/2000/09/xmldsig#`	[4]
`xades`	`http://uri.etsi.org/01903/v1.3.2#`	[2]
`ext`	`urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2`	UBL extension namespace

(continued)

Prefix	Namespace	Reference
sig	urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2	UBL signature extension apex namespace
sac	urn:oasis:names:specification:ubl:schema:xsd:SignatureAggregateComponents-2	UBL signature extension aggregate namespace
sbc	urn:oasis:names:specification:ubl:schema:xsd:SignatureBasicComponents-2	UBL signature extension basic namespace

8.4.3 Digital Signature Identification

This UBL extension is distinguished from other extensions and identified using the URI `urn:oasis:names:specification:ubl:dsig:enveloped` in the `<ext:ExtensionURI>` element.

NOTE In addition to Enveloped signatures, 8.3.3 Detached XML Signatures for UBL Documents also provides methods to be used with Detached signatures (i.e., digital signatures that stand outside the document being signed). Detached signatures constitute an independent technique without associated UBL artefacts, but an example instance showing detached signatures is included in this package; see 8.5 Digital Signature Examples.

8.4.4 Digital Signature Validation

The `UBL-ExtensionContentDataType-2.2.xsd` module links UBL validation to all needed extensions by importing the apex schema fragment of each extension vocabulary. The distribution version of this module supports IETF/W3C XML digital signatures by declaring that the `<ext:ExtensionContent>` element can contain elements from the UBL Digital Signature extension namespace. Accordingly, a single `<sig:UBLDocumentSignatures>` element is used as the apex of all the document's electronic signatures.

The `<ext:ExtensionContent>` element alternatively allows any other namespace apex element in order to allow other foreign extensions in the same document.

8.4.5 Digital Signature Structure

8.4.5.1 Digital Signature Structure Introduction

The signature extension structure exists to contain one or more IETF/W3C standard digital signature constructs. The UBL scaffolding for this extension starts with a `<ext:UBLExtension>` element with two children: `<ext:ExtensionURI>` (for extension distinction and identification) and `<ext:ExtensionContent>` (for containing the extension information, in this case the actual signatures and supporting information).

The signature extension Business Information Entities for UBL are contained in a single spreadsheet, provided here in two different formats.

mod/UBL-Signature-Entities-2.2.ods
mod/UBL-Signature-Entities-2.2.xls

An HTML rendition of the spreadsheet contents for the signature extension model also is provided:

mod/summary/reports/All-UBL-2.2-SignatureExtensionComponents.html

One or more signature extensions in a given document may each contain one or more sets of signature information. The following instructions guide the proper use of this particular extension.

8.4.5.2 Digital Signature Extension Metadata

The standard scaffolding for a given signature extension begins with the `<ext:UBLExtension>` element. The extension's role as a UBL signature extension is indicated with a child `<ext:ExtensionURI>` element with the `urn:oasis:names:specification:ubl:dsig:enveloped` value. The `urn:oasis:names:specification:ubl:dsig:enveloped:xades` value MAY be used to indicate the use of XAdES in the extension. Other extension metadata elements defined in UBL are allowed to be included for the convenience of users without changing the meaning or use of the extension.

```
<ext:UBLExtension>
  <ext:ExtensionURI
    >urn:oasis:names:specification:ubl:dsig:enveloped</ext:ExtensionURI>
  <ext:ExtensionContent>
```

8.4.5.3 The Extension Identifier

All uses of the optional <cbc:ID> metadata SHOULD be unique so that each extension can be uniquely identified. The identifier used can be any value. URNs beginning with urn:oasis:names:specification:ubl:extension: and ending with a number value are reserved for this purpose for the convenience of UBL users. The value urn:oasis:names:specification:ubl:extension:3 is an example of such a URN. As with all identifiers, each SHOULD be unique across all identifier values in a given UBL instance.

8.4.5.4 Digital Signature Apex

The mandatory <ext:ExtensionContent> element contains the UBL signature scaffolding. The apex element of the UBL signature information is <sig:UBLDocumentSignatures>.

8.4.5.5 Digital Signature Information

Each <sac:SignatureInformation> aggregate is used to contain the information related to a single IETF/W3C digital signature. Every signature added to the extension is isolated under a separate <sac:SignatureInformation> aggregate element containing the signature and its supporting information. As many of these aggregates can be in the extension as is needed, each one containing the information for a single digital extension.

```
  <ext:ExtensionContent>
    <sig:UBLDocumentSignatures>
      <sac:SignatureInformation>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

Note that three namespaces are used for signature information, in parallel with UBL's use of a document namespace, an aggregate namespace, and a basic namespace. The apex element is in the urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2 namespace, a parallel to a UBL document namespace. Signature-related aggregate entities are in the urn:oasis:names:specification:ubl:schema:xsd:SignatureAggregateComponents-2 namespace. Signature-related basic entities are in the urn:oasis:names:specification:ubl:schema:xsd:SignatureBasicComponents-2 namespace. Accordingly, there are three W3C Schema fragments in the distribution accommodating these three namespaces.

8.4.5.6 Digital Signature Identifier

An aggregate MAY be identified for referencing purposes using the common library <cbc:ID> element. Such an identifier may be useful in workflow scenarios where a particular signature needs to be identified external to the document, but its use is not obligatory. The identifier used can be any value. URNs beginning with urn:oasis:names:specification:ubl:signature: and ending with a number value are reserved for this purpose for the convenience of UBL users. The value urn:oasis:names:specification:ubl:signature:3 is an example of such a URN. As with all identifiers, each SHOULD be unique across all identifier values in a given UBL instance.

8.4.5.7 Digital Signature Reference

An aggregate MAY make reference to an existing <cac:Signature> business object in the same UBL document, but this is not obligatory. When needed, the <sbc:ReferencedSignatureID> basic element is used to point to the <cbc:ID> identifier value of the referenced <cac:Signature>. The identifier used can be any value. URNs beginning with urn:oasis:names:specification:ubl:signature: and ending with the local name of the parent of the signature business object, optionally followed with a colon and number, are reserved for this purpose for the convenience of UBL users. An example of such a URN is urn:oasis:names:specification:ubl:signature:IssuerEndorsement. As with all identifier references, the referenced identifier SHOULD exist and SHOULD be unique across all such identifier values in a given UBL instance.

See 8.3.2 Enveloped XML Signatures in UBL Documents for rules regarding common library UBL signature elements in the unextended portion of UBL documents that are being referenced by this element, together with an example of their use.

8.4.5.8 Digital Signature Content

A single <ds:Signature> element is a child of the aggregate. It MAY be absent from the document, thus supporting workflow scenarios where the element is added by a subsequent process after the UBL scaffolding is added by an earlier process. However, the signature information is semantically incomplete without the IETF/W3C-defined element. To support signatures countersigning this signature, this element must use the Id= attribute with a value unique among other attributes of schema type ID in the instance.

8.4.5.9 Example Digital Signature Skeleton

The following is a skeleton example of a single signature:

```
<ext:ExtensionContent>
  <sig:UBLDocumentSignatures
    xmlns:sig=
      "urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2"
    xmlns:sac=
      "urn:oasis:names:specification:ubl:schema:xsd:SignatureAggregateComponents-2"
    xmlns:sbc=
      "urn:oasis:names:specification:ubl:schema:xsd:SignatureBasicComponents-2">
    <sac:SignatureInformation>
      <cbc:ID>urn:oasis:names:specification:ubl:signature:1</cbc:ID>
      <sbc:ReferencedSignatureID
        >urn:oasis:names:specification:ubl:signature:Invoice
      </sbc:ReferencedSignatureID>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id=...>
        <ds:SignedInfo>
         ...
          <ds:Reference URI=...>
           ...
            <ds:Transform>
             ...
            </ds:Transform>
           ...
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
          ...
        </ds:SignatureValue>
        <ds:KeyInfo>
         ...
        </ds:KeyInfo>
        <ds:Object>
         ...
        </ds:Object>
      </ds:Signature>
    </sac:SignatureInformation>
  </sig:UBLDocumentSignatures>
</ext:ExtensionContent>
```

NOTE The XAdES specification contains all qualifying XAdES information in a single <ds:Object> element located as shown above. The UBL distribution includes and engages XAdES schema fragments with versions 1.3.2 and 1.4.1 for the convenience of users who choose to use these versions of XAdES. Users of the UBL signature extension are not obliged to use any XAdES extensions.

8.4.6 Transformation

The content to be signed is indicated in the URI= attribute of <ds:Reference>. Using the empty string indicates that the entire document (i.e. the enveloping UBL instance) is what is being signed:

```
<ds:Reference URI="">
```

A requirement when using digital signatures is to express in XPath that address that qualifies all nodes in the referenced content to be included in the calculation of the digital signature hash. For a signature added to a document to remain valid,

none of the information can change, nor can any information be added or removed from that portion of the document included in the hash calculation.

One of two such transformation expressions SHOULD be used in the UBL signature extension; users should choose the appropriate one to meet the objectives of adding the signature to the document. Adding non-signature information to the UBL document will invalidate all signatures already in the extension. The choice to make is whether to support additional signatures after adding the signature with the transformation expression.

The following transformation element in a digital signature flexibly prevents the signature from being invalidated by the subsequent addition of other signatures within the extension:

```
<Transform
     Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
   <XPath xmlns:sig=
     "urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2">
     count(ancestor-or-self::sig:UBLDocumentSignatures |
       here()/ancestor::sig:UBLDocumentSignatures[1]) >
         count(ancestor-or-self::sig:UBLDocumentSignatures)
   </XPath>
 </Transform>
```

The following transformation element in a digital signature is inflexible and thus would be considered a “final” signature to be added to the document. Such a signature will be invalidated by the subsequent addition of other signatures to the document:

```
<Transform
     Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
   <XPath xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     count(ancestor-or-self::ds:Signature |
       here()/ancestor::ds:Signature[1]) >
         count(ancestor-or-self::ds:Signature)
   </XPath>
 </Transform>
```

Multiple separate items of extra-document content (e.g., attachments) or embedded W3C signature content can be included in the same signature by using sibling `<ds:Reference>` elements with other `URI=` attribute values. For example, to countersign another signature in the same UBL document, make a local reference to that signature’s unique identifier, as in:

```
<ds:Reference URI="#{Id attribute of ds:Signature}">
```

NOTE To digitally sign only a portion of standard UBL content and not the entire document of UBL content, one uses an appropriate XPointer address for `URI=`. This requires XPointer awareness on the part of the digital signature tools being used.

8.5 Digital Signature Examples

The `xml/UBL-Invoice-2.0-Enveloped.xml` sample document illustrates the embedding of three extensions in a single document, one of which is a bona fide verifiable enveloped signature extension. A `<cac:Signature>` element makes reference to the embedded signature.

The `xml/UBL-Invoice-2.0-Detached.xml` sample document illustrates the placement of a detached digital signature outside of the UBL file. A `<cac:Signature>` element makes reference to the external signature.

The `xml/UBL-Invoice-2.0-Detached-Signature.xml` instance is an example of a bona fide verifiable digital signature of the `xml/UBL-Invoice-2.0-Detached.xml` instance.

9 Conformance

9.1 Document and Schema Conformance

The UBL 2.2 XSD schemas [5][6] are the only normative representations of the UBL 2.2 document types and library components for the purposes of XML document [3] validation and conformance.

An XML document is considered conforming to UBL 2.2 when all are true that:

1) there are no violations of the XSD validation schema constraints when using one of the normative document schemas listed in 6.2 UBL 2.2 Document Schemas,

2) there are no violations of the XSD constraints on extension scaffolding and metadata described in 6.5 Extension Methodology and Validation, and

3) there are no content violations of the constraints listed in 7 Additional Document Constraints.

NOTE Additional explanatory information regarding conformance as applied to UBL documents and schemas and their subsets, and the distinction between UBL conformance and UBL compatibility, is described in detail in the UBL 2 Guidelines for Customization [15]. That document has no bearing or impact on the clauses of this subsection.

9.2 Digital Signature Extension Conformance

9.2.1 Basic Digital Signature Extension Conformance

Claiming syntax conformance to the enveloped signature profile of UBL 2.2 requires the following:

— a schema-valid UBL extension in which the UBL Signature apex element is the apex of the extension;

— the `<ext:Extension>` element is present in the UBL extension and has either `urn:oasis:names:specification:ubl:dsig:enveloped` or `urn:oasis:names:specification:ubl:dsig:enveloped:xades` as its value;

— the value in all uses of `<sbc:ReferencedSignatureID>`, when present, correlates to a corresponding `<cbc:ID>` element of a `<cac:Signature>` element in the same instance; and

— the `<cbc:SignatureMethod>` element, when present, of signature business objects whose signatures are in the UBL extension has either `urn:oasis:names:specification:ubl:dsig:enveloped` or `urn:oasis:names:specification:ubl:dsig:enveloped:xades` as its value.

Claiming processing conformance to the enveloped profile of UBL 2.2 requires the conforming processing of all contained `<ds:Signature>` elements per [4].

Claiming syntax conformance to the detached profile of this specification requires that the `<cbc:SignatureMethod>` element, when present, of signature business objects whose signatures are outside of the UBL document has either `urn:oasis:names:specification:ubl:dsig:detached` or `urn:oasis:names:specification:ubl:dsig:detached:xades` as its value.

9.2.2 XAdES Digital Signature Extension Conformance

When conformance to XAdES in a UBL extension is chosen, UBL 2.2 requires the valid expression and processing of the XAdES syntax found in an XMLDSig per [2].

Annex A
(informative)

Release Notes

A.1 Availability

Online and downloadable versions of the latest OASIS release of this package are available from:

— http://docs.oasis-open.org/ubl/

Online and downloadable versions of the latest ISO/IEC release of this package are available from:

— http://standards.iso.org/ittf/PubliclyAvailableStandards/

A.2 Package Structure

The UBL 2.2 specification is published as a zip archive in the release directory. Unzipping this archive creates a directory named csprd02-UBL-2.2 containing a master DocBook XML file (UBL-2.2.xml), a generated hypertext version of this file (UBL-2.2.html), a generated PDF version of this file (UBL-2.2.pdf), and a number of subdirectories. The files in these subdirectories, linked to from UBL-2.2.xml, UBL-2.2.html, and UBL-2.2.pdf, contain the various normative and informational pieces of the 2.2 release. A description of each subdirectory is given below. Note that while the UBL-2.2.xml file is the "original" of this specification, it may not be viewable in all currently available web browsers.

art/

Diagrams and illustrations used in this specification

cl/

Code list specification files; see Annex E UBL 2.2 Code Lists and Two-phase Validation

cva/

Artefacts expressing data type qualifications; see [16] in 2 Normative References and Figure D.1 in Annex D Data Type Qualifications in UBL

db/

DocBook stylesheets for viewing UBL-2.2.xml

mod/

Spreadsheets and HTML renderings of the UBL data models; see Annex C The UBL 2.2 Data Model

val/

Test harness for demonstrating UBL 2.2 two-phase validation; see Annex E UBL 2.2 Code Lists and Two-phase Validation

xml/

Sample UBL 2.2 instances; see Annex F UBL 2.2 Example Document Instances

xsd/

XSD schemas; see 6 UBL 2.2 Schemas

xsdrt/

"Runtime" XSD schemas; see 6 UBL 2.2 Schemas

A.3 Support

UBL is a volunteer project of the international business community. Inquiries regarding UBL may be posted to the unmoderated public UBL-Dev list, archives for which are located at:

http://lists.oasis-open.org/archives/ubl-dev/

Subscriptions to UBL-Dev can be made through the OASIS list manager at:

http://www.oasis-open.org/mlmanage/index.php

OASIS provides an official community gathering place and information resource for UBL at:

http://ubl.xml.org/

The Wikipedia article for UBL has numerous related links:

http://www.wikipedia.org/wiki/Universal_Business_Language

A.4 UBL Customization

UBL provides a vocabulary that, for many user communities, can be used "as is". However, it is recognized that some user communities must address use cases whose requirements are not met by the UBL off-the-shelf solution. A separate OASIS Committee Specification known as the UBL 2 Guidelines for Customization [15] has been published to aid such users in developing custom solutions based on UBL.

The goal of UBL customization is to maintain a common understanding of the meaning of information being exchanged between specific implementations. The factors governing when to customize may be business-driven, technically driven, or both. The decision should be based on real-world needs balanced against perceived economic benefits.

A.5 Upgrading from UBL 2.0 or UBL 2.1 to UBL 2.2

For current UBL implementers, the most important thing to know about UBL 2.2 is that it is completely backward-compatible with UBL 2.0. In other words, any document that validates against a UBL 2.0 schema will validate against the UBL 2.2 version of that schema. The remaining differences relate mainly to the extended functionality that has been added to the 2.0 business processes in the areas of eTendering, sales reporting, utility statements, transport handling, and collaborative planning, forecasting, and replenishment (CPFR®).

Nonetheless, it would be unwise to simply overlay this UBL 2.2 release onto an existing installation, and the possible differences among existing installations are too large to allow a specific set of instructions to be provided for making the transition.

The brief history of UBL document types in the next section puts the new capabilities into context and may help users of existing UBL implementations decide whether to upgrade to 2.2.

New 2.2 users, on the other hand, can simply install 2.2 and rest assured that their software will interoperate with UBL documents generated by existing conforming UBL 2.0 installations. For more on the concept of conformance, see 9 Conformance and [15].

A.6 Known errors in UBL 2.2

During deployment the presence of errors in the UBL normative components comes to the attention of the UBL Technical Committee. Some of these cannot be repaired without breaking backwards compatibility to previous versions of UBL. Accordingly, they are obliged to remain in UBL untouched to avoid ambiguity and to avoid problems with backwards compatibility.

The list of known errors that are not being changed is as follows:

— the spelling of the BBIE named PartecipationPercent in the ABIE named ShareholderParty is incorrect

— the spelling of the BBIE named FirstShipmentAvailibilityDate in the ABIE named PromotionalEvent is incorrect

— the spelling of the BBIE named OccurenceLocation in the ABIE named Event is incorrect

— at this time there are no ASBIEs associating the common library ABIE with the DEN “Performance Data Line. Details”

Annex B
(informative)

Revision History

B.1 UBL Revisions

Since its first release as an OASIS Standard in 2004, UBL has experienced one major and now two minor version upgrades.

This appendix provides a description of the evolution of UBL.

B.2 UBL 1.0

Though apparently limited in scope, the eight document types provided in UBL 1.0 (2004) are applicable to a very large number of real-world use cases and have been widely deployed. These original 1.0 document types, later updated in UBL 2.0 and continued here in minor revisions, are Order, Order Response, Order Response Simple, Order Change, Order Cancellation, Despatch Advice, Receipt Advice, and Invoice. The figure below shows the original assumed process context for this most basic set of UBL document types. The scope of the process corresponds roughly to that of the UBL 2 Order, Fulfilment, and Traditional Billing processes described in the text (see 5.3.3.4 Ordering (post-award), 5.3.5.1 Logistics, and 5.3.7.1.3 Traditional Billing).

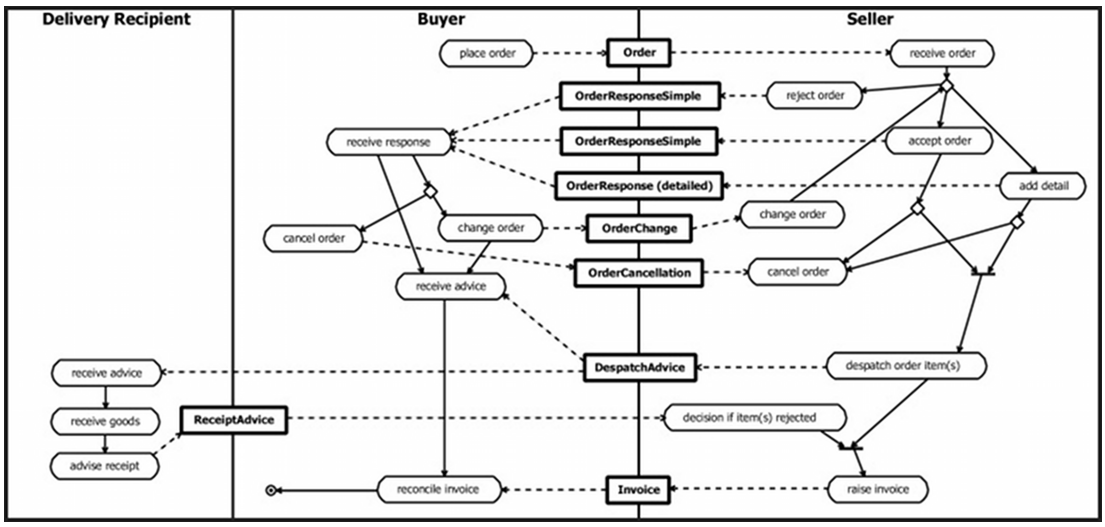

Figure B.1 — UBL 1.0 Order-to-Invoice Business Process

Because versions of UBL beginning with 2.0 do not maintain backward compatibility with UBL 1.0 document instances (that is, UBL 1.0 document instances will not validate against schemas from UBL 2.0 and later), use of UBL 1.0 in new installations is deprecated. Suitably revised versions of the original eight document types continue all the business functionality of UBL 1.0 in later versions.

B.3 Major Revision: UBL 2.0

Adoption of UBL 1.0 following ratification as an OASIS standard in November 2004 resulted in major inputs of new business content beyond the eight basic order-to-invoice business documents specified in the original release. In particular, contributions from representatives of government procurement, taxation, and transportation agencies in Europe, Asia, and North America resulted in greatly expanded pre-order and post-invoice capabilities together with the addition of several transport-related document types, bringing the total number of document types in UBL 2.0 to 31.

The new release also featured changes in UBL's use of XML schema methodology—most importantly, the adoption of global scoping for all element types—breaking backward compatibility with UBL 1.0 instances and therefore necessitating

designation as a major revision, signified by incrementing the version number from 1.0 to 2.0 rather than 1.1. The original eight UBL 1.0 document types were revised to reflect these changes.

UBL 2.0 achieved OASIS Standardization in December 2006, and the package was updated and corrected in May 2008.

The 23 document types added in UBL 2.0 can be summarized as follows:

Added UBL 2.0 document types for sourcing: Catalogue, Catalogue Deletion, Catalogue Item Specification Update, Catalogue Pricing Update, Catalogue Request, Quotation, Request For Quotation

Added UBL 2.0 document types for fulfilment: Bill Of Lading, Certificate Of Origin, Forwarding Instructions, Packing List, Transportation Status, Waybill

Added UBL 2.0 document types for billing: Credit Note, Debit Note, Freight Invoice, Reminder, Self Billed Credit Note, Self Billed Invoice

Added UBL 2.0 document types for payment: Remittance Advice, Statement

Added UBL 2.0 supplementary document types: Application Response, Attached Document

B.4 Minor Revision: UBL 2.1

Because it preserves backward compatibility with UBL 2.0, UBL 2.1 is technically a minor release, not a major one. However, it did add 34 new document types, bringing the total number of UBL business documents to 65.

Added UBL 2.1 document types for eTendering: Awarded Notification, Call For Tenders, Contract Award Notice, Contract Notice, Guarantee Certificate, Tender, Tender Receipt, Tenderer Qualification, Tenderer Qualification Response, Unawarded Notification

Added UBL 2.1 document types for Collaborative planning, forecasting, and replenishment: Exception Criteria, Exception Notification, Forecast, Forecast Revision, Item Information Request, Prior Information Notice, Trade Item Location Profile

Added UBL 2.1 document types for Vendor Managed Inventory: Instruction For Returns, Inventory Report, Product Activity, Retail Event, Stock Availability Report

Added UBL 2.1 document types for fulfilment: Fulfilment Cancellation

Added UBL 2.1 document types for Intermodal Freight Management: Goods Item Itinerary, Transport Execution Plan, Transport Execution Plan Request, Transport Progress Status, Transport Progress Status Request, Transport Service Description, Transport Service Description Request, Transportation Status, Transportation Status Request

Added UBL 2.1 document type for Utility billing: Utility Statement

Added UBL 2.1 supplementary document types: Document Status, Document Status Request

The 8 UBL Digital Signatures extension was added in UBL 2.1. This extension works as is also with UBL 2.0.

Details of the changes from UBL 2.0 to UBL 2.1 are found at http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html#S-MINOR-REVISION-UBL-2.1

B.5 Minor Revision: UBL 2.2

B.5.1 New Document Types in UBL 2.2

Because it preserves backward compatibility with UBL 2.1 and UBL 2.0, UBL 2.2 is technically a minor release, not a major one. However, it did add 16 new document types, bringing the total number of UBL business documents to 81.

Added UBL 2.2 document types for eTendering: Enquiry, Enquiry Response, Expression Of Interest Request, Expression Of Interest Response, Qualification Application Request, Qualification Application Response, Tender Contract, Tender Status, Tender Status Request, Tender Withdrawal, Unsubscribe From Procedure Request, Unsubscribe From Procedure Response

Added UBL 2.2 document types for transportation: Weight Statement

Added UBL 2.2 document types for business directories and agreements: Business Card, Digital Agreement, Digital Capability

B.5.2 Schema changes from UBL 2.1 to UBL 2.2

B.5.2.1 Schema Changes Introduction

The following two tables show the differences between the XML elements in UBL 2.1 and those in UBL 2.2.

All changes in 2.2 schemas are backward-compatible with valid UBL 2.1 and UBL 2.0 instances. Changes include the addition of new elements and attributes; changes in cardinality from 1 to 0..1 (i.e., making a formerly required element optional); changes in cardinality from 0..1 to 0..n (i.e., allowing an unlimited number of occurrences instead of just one); and corrections to Dictionary Entry Names (DENs).

B.5.2.2 Changes to Library Elements, UBL 2.1 to UBL 2.2

The following table sums up the differences between the XML elements in the UBL 2.1 Common Library and those in the UBL 2.2 Common Library.

Aggregate BIE	Basic or Association BIE	Changes for UBL
Attachment		
	EmbeddedDocument	Added
AwardingTerms		
	NoFurtherNegotiationIndicator	Added
Capability		
	WebSite	Added
ClassificationScheme		
	AgencyName	Changed dictionary entry name from “Classification Scheme. Agency Name. Text” to “Classification Scheme. Agency Name. Name”
Consignment		
	ActualPickupTransportEvent	Added
	ActualDeliveryTransportEvent	Added
ContractExtension		
	RenewalsIndicator	Added
ContractingSystem		Added
DeliveryChannel		Added
DigitalAgreementTerms		Added
DigitalCollaboration		Added
DigitalProcess		Added
DigitalService		Added
DocumentDistribution		
	DocumentTypeCode	Added
	MaximumCopiesNumeric	Changed cardinality from 1 to 0..1
	MaximumOriginalsNumeric	Added
DocumentMetadata		Added
EconomicOperatorParty		Added
EncryptionCertificatePathChain		Added
EncryptionData		Added

(continued)

Aggregate BIE	Basic or Association BIE	Changes for UBL
EncryptionSymmetricAlgorithm		Added
Evidence		
	Name	Added
	ConfidentialityLevelCode	Added
	DocumentReference	Changed cardinality from 0..1 to 0..n
Legislation		Added
LotDistribution		Added
MessageDelivery		Added
Meter		
	MeterName	Changed dictionary entry name from “Meter. Meter Name. Text” to “Meter. Meter Name. Name”
MonetaryTotal		
	WithholdingTaxTotalAmount	Added
ParticipantParty		Added
Party		
	AdditionalWebSite	Added
	SocialMediaProfile	Added
Person		
	BirthplaceName	Changed dictionary entry name from “Person. Birthplace Name. Text” to “Person. Birthplace Name. Name”
	RoleCode	Added
	CitizenshipCountry	Added
PostAwardProcess		Added
ProcurementProject		
	Name	Changed cardinality from 1..n to 0..n
	MainCommodityClassification	Changed cardinality from 0..1 to 0..n
ProcurementProjectLotReference		Added
ResponseValue		Added
ServiceLevelAgreement		Added
SocialMediaProfile		Added
TenderPreparation		
	TenderEncryptionData	Added
TenderResult		
	AwardID	Added
	ReceivedTenderQuantity	Changed dictionary entry name from “Tender Result. Received_ Tender. Quantity” to “Tender Result. Received_ Tender Quantity. Quantity”
	LowerTenderAmount	Changed dictionary entry name from “Tender Result. Lower_ Tender. Amount” to “Tender Result. Lower_ Tender Amount. Amount”
	HigherTenderAmount	Changed dictionary entry name from “Tender Result. Higher_

(continued)

Aggregate BIE	Basic or Association BIE	Changes for UBL
		Tender. Amount" to "Tender Result. Higher_ Tender Amount. Amount"
TenderingCriterion		Added
TenderingCriterionProperty		Added
TenderingCriterionPropertyGroup		Added
TenderingCriterionResponse		Added
TenderingProcess		
	AccessToolsURI	Added
	EconomicOperatorShortList	Changed cardinality from 0..1 to 0..n
	ContractingSystem	Added
TenderingTerms		
	RecurringProcurementIndicator	Added
	EstimatedTimingFurtherPublication	Added
	LotDistribution	Added
	PostAwardProcess	Added
	EconomicOperatorShortList	Added
TransportEquipment		
	VerifiedGrossMass	Added
VerifiedGrossMass		Added
WebSite		Added

B.5.2.3 Changes to Document Elements, UBL 2.1 to UBL 2.2

The following table sums up the differences between the XML elements in the UBL 2.1 document schemas and those in the UBL 2.2 document schemas.

Aggregate BIE	Basic or Association BIE	Changes for UBL
AwardedNotification		
	ContractName	Changed dictionary entry name from "Awarded Notification. Contract Name. Text" to "Awarded Notification. Contract Name. Name"
BusinessCard		Added
CallForTenders		
	ContractingParty	Changed cardinality from 1 to 1..n
ContractAwardNotice		
	NoticeLanguageCode	Added
	ContractingParty	Changed cardinality from 1 to 1..n
ContractNotice		
	NoticeTypeCode	Added
	NoticeLanguageCode	Added
	ContractingParty	Changed cardinality from 1 to 1..n
CreditNote		
	DueDate	Added
	ProjectReference	Added
	WithholdingTaxTotal	Added

(continued)

Aggregate BIE	Basic or Association BIE	Changes for UBL
DebitNote		
	WithholdingTaxTotal	Added
DigitalAgreement		Added
DigitalCapability		Added
Enquiry		Added
EnquiryResponse		Added
ExpressionOfInterestRequest		Added
ExpressionOfInterestResponse		Added
ForwardingInstructions		
	DocumentDistribution	Added
FreightInvoice		
	DueDate	Added
	ProjectReference	Added
	WithholdingTaxTotal	Added
GuaranteeCertificate		
	Signature	Changed cardinality from 1..n to 0..n
OrderResponse		
	OrderChangeDocumentReference	Added
OrderResponseSimple		
	OrderChangeDocumentReference	Added
PriorInformationNotice		
	NoticeTypeCode	Added
	NoticeLanguageCode	Added
	ContractingParty	Changed cardinality from 1 to 1..n
QualificationApplicationRequest		Added
QualificationApplicationResponse		Added
SelfBilledCreditNote		
	DueDate	Added
	CreditNoteTypeCode	Added
	BuyerReference	Added
	ProjectReference	Added
	WithholdingTaxTotal	Added
SelfBilledInvoice		
	DueDate	Added
	BuyerReference	Added
	ProjectReference	Added
	WithholdingTaxTotal	Added
Tender		
	ContractName	Changed dictionary entry name from “Tender. Contract Name. Text” to “Tender. Contract Name. Name”
	TendererParty	Changed cardinality from 1 to 1..n
	ContractingParty	Changed cardinality from 0..1 to 0..n
TenderContract		Added
TenderReceipt		

(continued)

Aggregate BIE	Basic or Association BIE	Changes for UBL
	ContractName	Changed dictionary entry name from “Tender Receipt. Contract Name. Text” to “Tender Receipt. Contract Name. Name”
TenderStatus		Added
TenderStatusRequest		Added
TenderWithdrawal		Added
TendererQualificationResponse		
	ContractName	Changed dictionary entry name from “Tenderer Qualification Response. Contract Name. Text” to “Tenderer Qualification Response. Contract Name. Name”
UnawardedNotification		
	ContractName	Changed dictionary entry name from “Unawarded Notification. Contract Name. Text” to “Unawarded Notification. Contract Name. Name”
UnsubscribeFromProcedureRequest		Added
UnsubscribeFromProcedureResponse		Added
WeightStatement		Added

B.5.3 Editorial changes from UBL 2.1 to UBL 2.2

As this is a very lengthy specification, this guidance to the reader reflects where UBL 2.2 has not changed substantially or substantively from UBL 2.1. Editorial changes that are related to grammar, spelling and turn of phrase are not enumerated.

0 Introduction is unchanged from UBL 2.1 with the exception of citing the intended primary audiences for this specification.

5 UBL 2.2 Business Objects has been augmented with an overall view diagram and information regarding a number of subject areas. No subject areas from UBL 2.1 have been removed from this section.

6 UBL 2.2 Schemas has been augmented with a number of new document types and references to example instances. 6.3.4 Extension Content Schemas is modified for clarity regarding the user’s latitude when adding extensions.

7 Additional Document Constraints is unchanged from UBL 2.1 with the exception of the addition of an explanatory comment regarding [IND5] and [IND6]. No constraints have been changed or added.

8 UBL Digital Signatures is unchanged from UBL 2.1 with the exception of the importation of updated XAdES schema fragments in the extension content schema fragment.

9 Conformance is unchanged from UBL 2.1 with the exception of calling out from an external document into this document the applicable information regarding schema and content conformance.

Annex A Release Notes is unchanged with the exception of adding UBL 2.2 to the section on upgrading, and enumerating the known errors in the document models.

Annex B Revision History summarizes the changes from UBL 2.0 to UBL 2.1 and details the changes from UBL 2.1 to UBL 2.2. Other sections are unchanged.

Annex C The UBL 2.2 Data Model is largely unchanged from UBL 2.1 with the exception of file references, line numbers and adding hyperlinks to the model reports. Some information previously found in separate sub-clauses has been consolidated into the first sub-clause. References to UML diagrams have been removed.

Annex D Data Type Qualifications in UBL is unchanged from UBL 2.1 with the exception of a revised diagram and referencing the UBL 2.1 release.

Annex E UBL 2.2 Code Lists and Two-phase Validation is unchanged from UBL 2.1 with the exception of the list of code lists.

Annex F UBL 2.2 Example Document Instances includes a revised list of example instances.

Annex G Alternative Representations of the UBL 2.2 Schemas is revised to reference only a free RELAX-NG tool with which to convert the normative UBL schemas into an alternative syntax.

Annex H The Open-edi reference model perspective of UBL is unchanged.

Annex I Acknowledgements is changed to reflect the active membership of the technical committee during the development of UBL 2.2.

B.5.4 Removal of non-normative artefacts

The UBL 2.2 release does not include the non-normative RELAX-NG and UML diagram alternative representations of the UBL normative schemas that are found in earlier releases.

B.5.5 Temporary Annex: Schema changes from UBL 2.2 csprd01 to UBL 2.2 csprd02

B.5.5.1 UBL 2.2 csprd01 Schema Changes Introduction

The following two tables show the differences between the XML elements in UBL 2.1 and those in UBL 2.2.

All changes in 2.2 schemas are backward-compatible with valid UBL 2.1 and UBL 2.0 instances. Changes include the addition of new elements and attributes; changes in cardinality from 1 to 0..1 (i.e., making a formerly required element optional); changes in cardinality from 0..1 to 0..n (i.e., allowing an unlimited number of occurrences instead of just one); and corrections to Dictionary Entry Names (DENs).

B.5.5.2 Changes to Library Elements, UBL 2.2 csprd01 to UBL 2.2 csprd02

The following table sums up the differences between the XML elements in the UBL 2.2 csprd01 Common Library and those in the current UBL 2.2 csprd02 Common Library.

Aggregate BIE	Basic or Association BIE	Changes for UBL
EconomicOperatorParty		
	QualifyingParty	Added
Evidence		
	ConfidentialityLevelCode	Added
Legislation		
	Language	Changed dictionary entry name from “Legislation. Language. Text” to “Legislation. Language” and changed component type from BBIE to ASBIE
MonetaryTotal		
	WithholdingTaxTotalAmount	Added
ResponseValue		
	ResponseURI	Added
TenderingCriteriaGroup		Deleted
TenderingCriterion		
	TenderingCriteriaGroup	Deleted
	TenderingCriterionPropertyGroup	Added
TenderingCriterionProperty		Added
TenderingCriterionPropertyGroup		Added
TenderingCriterionResponse		
	ValidatedCriterionRequirementID	Deleted

(continued)

Aggregate BIE	Basic or Association BIE	Changes for UBL
	Evidence	Deleted
	ValidatedCriterionPropertyID	Added
	ConfidentialityLevelCode	Added
	EvidenceSupplied	Added

B.5.5.3 Changes to Document Elements, UBL 2.2 csprd01 to UBL 2.2 csprd02

The following table sums up the differences between the XML elements in the UBL 2.2 csprd01 document schemas and those in the current UBL 2.2 csprd02 document schemas.

Aggregate BIE	Basic or Association BIE	Changes for UBL
BusinessCard		
	SenderParty	Changed cardinality from 1 to 0..1
	ReceiverParty	Changed cardinality from 1 to 0..1
	BusinessParty	Changed cardinality from 0..1 to 1
CreditNote		
	DueDate	Added
	ProjectReference	Added
DigitalCapability		
	SenderParty	Changed cardinality from 1 to 0..1
	ReceiverParty	Changed cardinality from 1 to 0..1
	BusinessParty	Changed cardinality from 0..1 to 1
FreightInvoice		
	DueDate	Added
	ProjectReference	Added
QualificationApplicationRequest		
	ProcedureCode	Added
	QualificationApplicationTypeCode	Added
	WeightScoringMethodologyNote	Added
	WeightingTypeCode	Added
	EconomicOperatorParty	Changed cardinality from 0..1 to 0..n
QualificationApplicationResponse		
	TendererRoleCode	Deleted
	ProcedureCode	Added
	QualificationApplicationTypeCode	Added
	WeightScoringMethodologyNote	Added
	WeightingTypeCode	Added
	EconomicOperatorParty	Changed cardinality from 1 to 1..n
	TenderingCriterionResponse	Changed cardinality from 0..n to 1..n
	Evidence	Added
SelfBilledCreditNote		
	DueDate	Added
	CreditNoteTypeCode	Added
	BuyerReference	Added
	ProjectReference	Added
SelfBilledInvoice		

(continued)

Aggregate BIE	Basic or Association BIE	Changes for UBL
	DueDate	Added
	BuyerReference	Added
	ProjectReference	Added

Annex C
(informative)

The UBL 2.2 Data Model

C.1 The Use of the OASIS Business Document Naming and Design Rules

As described in the OASIS UBL Naming and Design Rules [28] application of the OASIS Business Document Naming and Design Rules [10], the UBL data model design follows the principles of the UN/CEFACT Core Components Technical Specification [12]. The UBL data model is based on a library of reusable information items known as Business Information Entities (BIEs). Each business document defined by UBL is created by assembling items appropriate to that document type from the UBL BIE library. Further detail regarding BIEs is provided in C.4 Business Information Entities.

Historically, both the UBL common library of reusable components and the assembly models for the individual UBL documents have been published as separate spreadsheets using a format specifically developed for UBL business information modeling (this format is discussed further below). Beginning with UBL 2.2, all of these models are published as separate worksheets in a single spreadsheet. This spreadsheet is provided in both Open Document and Microsoft Excel formats in `mod/` subdirectory:

`mod/UBL-Entities-2.2.ods`
`mod/UBL-Entities-2.2.xls`

A machine-processable XML version of the spreadsheet contents for the entire UBL data model is provided in OASIS genericode [20] format:

`mod/UBL-Entities-2.2.gc`

Similar files for the UBL standardized signature extension also are in `mod/` subdirectory:

`mod/UBL-Signature-Entities-2.2.ods`
`mod/UBL-Signature-Entities-2.2.xls`
`mod/UBL-Signature-Entities-2.2.gc`

An HTML rendition of the spreadsheet contents for the entire UBL data model and of the signature extension data model are provided:

`mod/summary/reports/All-UBL-2.2-Documents.html`
`mod/summary/reports/All-UBL-2.2-SignatureExtensionComponents.html`

For links to the individual HTML reports for each of the document types, see the schema tables in 6.2 UBL 2.2 Document Schemas. These reports elide all of the library components that are not used by each document type and are far shorter than the “all documents” report.

For notes on the use of the HTML reports, see

`mod/summary/readme-Reports.html`

C.2 UBL Validation Artefact Generation

Following the relevant sections of the OASIS Business Document Naming and Design Rules, the normative UBL schemas and non-normative OASIS Context/Value Association [16] file are generated from the machine-processable XML of the spreadsheet contents. From the CVA file and the genericode expressions of code list values, the data type qualifications XSLT stylesheet is generated.

The following diagram shows the conceptual relationships between the UBL data models on the left and validation artefacts (schemas and XSLT) on the right. Compare Figure 78.

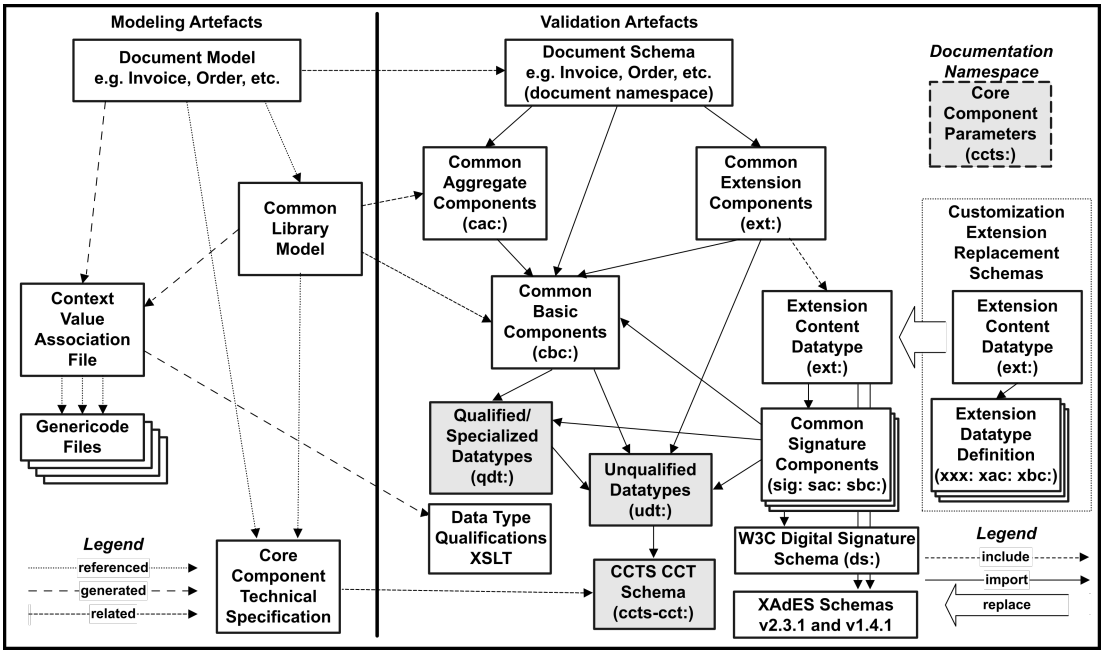

Figure C.1 — UBL Data Model Realization

The namespaces shown in the shaded boxes (with prefixes `qdt:`, `udt:`, `ccts-cct:` and `ccts:`) exist for the management of the schema components only and have no utility in UBL XML document instances. Declaring unused namespaces in an XML instance is superfluous and does not impact on conformance, but having them present may be confusing or misleading to the reader.

C.3 The UBL Common Library

As noted above, UBL is based on a reusable library of Business Information Entities. In the current release, the Common Library contains more than two thousand of these individually defined data items.

C.4 Business Information Entities

In the language of [12], UBL Business Information Items (BIEs) include BBIEs ("basic" individual pieces of information), ABIEs (aggregations of other BIEs), and ASBIEs ("associations" to ABIEs). Fuller explanations of these terms in the context of the CCTS framework will be found in the CCTS specification. For purposes of understanding UBL as a set of XML schemas, however, it may be useful to describe these terms employing concepts more familiar to XML users.

With the understanding that every XML document describes a logical tree of elements, the different kinds of Business Information Entities from which UBL documents are constructed may be described as follows:

UBL BBIEs (Basic Business Information Entities) are the leaf nodes of every UBL document structure. These are ordinary data fields such as one would expect to find in any business form, and they are realized in the schemas as individual XML elements at the bottom level of the document tree with simple content representing amounts, codes, quantities, and so on. All UBL BBIE elements (and only UBL BBIE elements) are members of the UBL common basic components namespace, conventionally denoted in UBL schemas by the `cbc:` prefix. (Since all namespace prefixes in XML are assigned on a per-instance basis according to namespace declarations in the individual instance, prefixes such as `cbc:` may be replaced with arbitrarily different namespace prefixes in actual UBL documents.)

UBL ASBIEs (Association Business Information Entities) are substructures of a UBL document. Children of ASBIEs may be BBIEs or other ASBIEs, never ABIEs. All UBL ASBIEs (and only UBL ASBIEs) are members *as elements* of the UBL common aggregate components namespace, denoted in UBL schemas by the `cac:` prefix.

UBL document ABIEs (Aggregate Business Information Entities) are the root nodes and top-level structures of UBL documents. Children of document ABIEs may be BBIEs or ASBIEs, never ABIEs. All UBL document ABIEs (and only UBL document ABIEs) are defined within individual namespaces specific to each document as both elements and types.

UBL library ABIEs (that is, all ABIEs except document ABIEs) have a structural shape but are not concrete document structures; rather, they are abstract structures or templates for ASBIEs, thus allowing the same structure to be reused in multiple roles. Children of library ABIEs in the data structure can be BBIEs or ASBIEs, never ABIEs. All library ABIEs must be realized as ASBIEs in order to actually exist as elements in the UBL document tree. All UBL library ABIEs (and only UBL library ABIEs) are realized *as types* in the UBL `cac:` namespace.

This naming scheme inherited from CCTS may prove problematic for some UBL users. In particular, the CCTS terms "Association Business Information Entity" and "Aggregate Business Information Entity" do not well describe these two concepts as they are realized in XML. The problem word here is "association", which correctly describes this relationship within a UML (Unified Modeling Language) framework but is perhaps better thought of in the UBL context as meaning that a particular ASBIE is "associated with" an abstract ABIE structure. For our purposes, it would have been better if ASBIEs had instead been called "Aggregate Business Information Entities" and ABIEs had instead been called "Structural Templates". It may prove easiest for the UBL user to regard the terms ASBIE and ABIE as opaque labels and to ignore the historical expansions of these acronyms.

It can be seen from the above that the XML implementations of ASBIEs and library ABIEs share the same `cac:` namespace. In the schemas, library ABIEs are all implemented as XML types, and ASBIEs are all implemented as XML elements. This is simply a reflection of their different roles—library ABIEs as abstract classes or structural templates (realized as XML types) and ASBIEs as concrete instantiations (realized as XML elements derived from those types).

While the distinction between ABIEs/classes/types on the one hand and ASBIEs/instantiations/elements on the other is clear enough, it should be noted that in some cases an ASBIE does not qualify the name of the ABIE from which it is derived. In effect, they have the same name. Some library ABIEs are used only in the form of an ASBIE having the same name; for example, `AddressLine` is a library ABIE that is only used in the form of an ASBIE named `AddressLine`. Some library ABIEs are realized in some places as ASBIEs with the same name (where it is felt that the unqualified name is sufficient) and elsewhere as ASBIEs with a name that is further qualified; for example, the library ABIE `Address` has numerous ASBIE realizations with qualified names like `LocationAddress`, `ApplicableAddress`, `DespatchAddress`, and so on, but it's also seen as an ASBIE simply named `Address` that's included in the library ABIEs `FinancialInstitution`, `Branch`, `Location`, and `ConsumptionPoint`. Some library ABIEs are never actually implemented as ASBIEs with the same name; for example, only one ASBIE is associated with the library ABIE `ActivityDataLine`, and it has the qualified name `SupplyChainActivityDataLine`.

The UBL Common Aggregate Component schema declares an identically named element or potential ASBIE for every library ABIE regardless of whether that element is used in a UBL document schema to represent an ASBIE (these are among the long list of global element declarations at the beginning of the CAC module). ABIEs are implemented as one or more ASBIEs via XSD references to these elements farther down in the CAC schema module or in individual document schema modules, which all import the CAC module. For example, the global element `AddressLine` declared in the CAC with the line

```
<xsd:element name="AddressLine" type="AddressLineType"/>
```

is implemented as an ASBIE with the same name in the declaration of the `Address` ABIE as follows:

```
<xsd:element ref="cac:AddressLine" minOccurs="0" maxOccurs="unbounded">
   [...]
</xsd:element>
```

One consequence of this approach is that the list of global elements that begins the CAC module contains elements that are in fact never used under those names in UBL 2.2. For example, the element `ActivityDataLine` mentioned above is used by reference in creating the ASBIE `SupplyChainActivityDataLine`, but it never appears in the form of an ASBIE named `ActivityDataLine`. Such unused ABIE names remain available in the global element declarations for customizers and designers of future additions to UBL.

C.5 Navigating the UBL Data Model

The concepts described above can be illustrated by navigating the UBL data model to construct a trivial UBL Invoice instance.

We will start with a wrapper copied from an example in the xml/ directory of the UBL distribution (xml/UBL-Invoice-2.1-Example.xml) that has the required XML namespace declarations for the Invoice and for the common library components ("cac" for the aggregate (ABIE and ASBIE) components and "cbc" for the basic (BBIE) components):

```
<?xml version="1.0" encoding="UTF-8"?>
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2">
[...]
</Invoice>
```

Now we will fill out this shell of an instance, completing the part in the square brackets by traversing the data model.

In addition to the aforementioned complete UBL data model spreadsheets and HTML rendering, when dealing with only a single document type there is an HTML rendition of that subset of the spreadsheet contents with only that content utilized by the one document type, such as for the Invoice:

mod/summary/reports/UBL-Invoice-2.2.html

Line 2 of the Invoice model defines the document ABIE named Invoice. The Component Type column confirms that Invoice is an ABIE, as also indicated by the pink background in that row of the rendering.

Everything after Invoice in the model ends up as part of the schema, and the order seen here is the order in which these components will appear in both the schema and any conforming instances of Invoice. The BBIE children of Invoice are given first (white background), and then all the ASBIE children of Invoice (green background).

As shown in Cardinality column, most of these components are optional. The first required field is ID (line 7) and the second is IssueDate (line 10), so we can write, for example,

```
<cbc:ID>123</cbc:ID>
<cbc:IssueDate>2011-09-22</cbc:IssueDate>
```

Next let's add an optional InvoicePeriod (line 25). This is an ASBIE, implying that it has some kind of substructure, and it derives from the generic ABIE called Period (this is the "Associated Object Class" referred to in a column of the same name). To find this structure, we look for the Period library ABIE in the model report or in the Common Library worksheet of the UBL model spreadsheet.

Period will be found at line 1510 and seen to contain a number of possible BBIE children, all of them optional; and the ASBIE InvoicePeriod in Invoice therefore has this structure, too. From this one could conclude that instantiations of the Period structure (there are more than 50 of them in UBL) need not contain any of the seven optional BBIE elements specified after line 1510, and indeed the corresponding declaration of the complex type PeriodType in the CAC schema (xsd/common/UBL-CommonAggregateComponents-2.2.xsd) shows that an empty InvoicePeriod element will pass XML validation; but UBL explicitly prohibits such structures (see 7.4 Empty Elements). In UBL, as a normative rule independent of schema constraints, every ASBIE must have at least one child (BBIE or ASBIE) instantiated. In this case, therefore, one or more of the seven possible BBIE children of InvoicePeriod will need to appear in a UBL Invoice document for it to be conforming to UBL in addition to the requirement that the document validate against the Invoice schema. If StartDate and EndDate (for exsample) are chosen for the content of InvoicePeriod, the corresponding section of the sample instance might then look like this:

```
<cac:InvoicePeriod>
  <cbc:StartDate>2011-08-01</cbc:StartDate>
  <cbc:EndDate>2011-08-31</cbc:EndDate>
</cac:InvoicePeriod>
```

Next in order in the Invoice come two required pieces, the ASBIEs AccountingSupplierParty and AccountingCustomerParty. As shown in Associated Object Class column of the Invoice model, AccountingSupplierParty (line 36) derives from the SupplierParty ABIE and AccountingCustomerParty (line 37) derives from the CustomerParty ABIE. Checking in the Common Library, it is seen that both SupplierParty (line 2039 of the Common Library) and CustomerParty (line 562 of the Common Library) can contain an ASBIE named Party (as shown in lines 2043 and 566, respectively) and that each Party ASBIE is an instantiation of the Party ABIE (line 1402). Therefore both parties have the same structure (the BBIEs and ASBIEs following line 1402). Thus AccountingSupplierParty and AccountingCustomerParty share the information components common to parties in general and differ in the information specific to suppliers and customers. Parties commonly have a PartyName (line 1410)

that derives (the Associated Object Class column) from the ABIE PartyName (line 1441), which is a wrapper for the BBIE Name (line 1442). A conforming piece of the document instance might therefore look like this:

```
<cac:AccountingSupplierParty>
  <cac:Party>
    <cac:PartyName>
      <cbc:Name>Custom Cotter Pins</cbc:Name>
    </cac:PartyName>
  </cac:Party>
</cac:AccountingSupplierParty>
<cac:AccountingCustomerParty>
  <cac:Party>
    <cac:PartyName>
      <cbc:Name>North American Veeblefetzer</cbc:Name>
    </cac:PartyName>
  </cac:Party>
</cac:AccountingCustomerParty>
```

Returning to the Invoice model, it is seen that the Invoice must close with a LegalMonetaryTotal (line 54) and at least one InvoiceLine (line 55). Taking LegalMonetaryTotal first, it is found in the Common Library to be derived from MonetaryTotal (line 1324), which has a mandatory PayableAmount BBIE. A corresponding example instance fragment might be therefore be constructed as follows:

```
<cac:LegalMonetaryTotal>
   <cbc:PayableAmount currencyID="CAD">100.00</cbc:PayableAmount>
</cac:LegalMonetaryTotal>
```

If the preceding explanation of Party is understood, there should be nothing problematic about the process of forming the example LegalMonetaryTotal element shown above except the currencyID attribute on PayableAmount, which does not appear explicitly in the model line for that BBIE (line 1333). This is because UBL does not define the primitive data types upon which the model is built; instead it uses standard data type definitions from [12] and [6]. In the case of PayableAmount, the CCTS data type (the Data Type column) is “Amount. Type” (the space is part of the name), and that type is defined in [12] itself (Table 8-1 of the CCTS 2.01 specification). There it will be seen that “Amount. Type” has two supplementary “CCT Components” called “Amount. Currency. Identifier” and “Currency. Code List Version. Identifier”. In the XML realization of CCTS, supplementary components are expressed as attributes, and the CCTS names “Amount. Currency. Identifier”and “Currency. Code List Version. Identifier” are transformed into the XML attribute names currencyID and currencyCodeListVersionID, respectively. All of these CCTS-based types and attributes are declared in the CCTS Core Component Type schema module:

xsd/common/CCTS_CCT_SchemaModule-2.2.xsd

Note that this schema module comes from UN/CEFACT, not UBL; that it does not implement all of the supplementary components of core component types defined by [12]; and that all of the attributes it does declare are defined as optional. In UBL, however, the attributes currencyID and mimeCode are required, not optional. In order to impose its own restrictions, therefore, and also to supply a full set of supplementary component attributes, UBL provides an Unqualified Data Types module that imports the CCTS module and then overrides those definitions as needed:

xsd/common/UBL-UnqualifiedDataTypes-2.2.xsd

Further information about UBL data types can be found in Annex D Data Type Qualifications in UBL. Note in particular Table D.2, which includes a list of all the attributes associated with UBL unqualified data types. A reverse lookup of the implied occurrence of each attribute in the data models is provided in this summary report:

mod/summary/reports/All-UBL-2.2-Documents.html#UDT

In the example fragment above, currencyID has been used to label the amount in Canadian dollars (CAD). As explained in Annex E UBL 2.2 Code Lists and Two-phase Validation, the value CAD for this attribute is not specified in schemas to be checked using XSD validation but will instead be found in separate OASIS genericode code list files in the gc/ directory of the UBL distribution, which are engaged through a separate XSLT-based process.

Using the same methodology, a sample InvoiceLine can be constructed to complete the example as follows:

```
<cac:InvoiceLine>
  <cbc:ID>1</cbc:ID>
```

```
    <cbc:LineExtensionAmount currencyID="CAD">100.00</cbc:LineExtensionAmount>
    <cac:Item>
       <cbc:Description>Cotter pin, MIL-SPEC</cbc:Description>
    </cac:Item>
  </cac:InvoiceLine>
```

The finished example can be found in

xml/UBL-Invoice-2.1-Example-Trivial.xml

Annex D
(informative)

Data Type Qualifications in UBL

All UBL data types ultimately derive either from the UN/CEFACT Core Components Technical Specification [12] Core Component Types (CCT) or from the W3C Schema specification [6] itself; this derivation takes place in the UBL UDT module. The following table lists the CCTS 2.01 Core Component Types.

CCTS Data Type	Definition
Amount. Type	A number of monetary units specified in a currency where the unit of currency is explicit or implied.
Binary Object. Type	A set of finite-length sequences of binary octets.
Code. Type	A character string (letters, figures or symbols) that for brevity and/or language independence may be used to represent or replace a definitive value or text of an Attribute together with relevant supplementary information.
Date Time. Type	A particular point in the progression of time together with relevant supplementary information.
Identifier. Type	A character string to identify and distinguish uniquely, one instance of an object in an identification scheme from all other objects in the same scheme together with relevant supplementary information.
Indicator. Type	A list of two mutually exclusive Boolean values that express the only possible states of a Property.
Measure. Type	A numeric value determined by measuring an object along with the specified unit of measure.
Numeric. Type	Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.
Quantity. Type	A counted number of non-monetary units possibly including fractions.
Text. Type	A character string (i.e. a finite set of characters) generally in the form of words of a language.

The UBL unqualified data types include the CCTS unqualified data types (named according to the UBL Naming and Design Rules) and a few others, as listed in the following table. Some of these (`GraphicType`, `PictureType`, `SoundType`, `VideoType`, and `ValueType`) are defined for completeness but not actually used in UBL 2.2.

The rightmost column of this table lists the UBL XML attributes that implement the CCTS supplementary components associated with each CCTS data type. It is important to be aware of these attributes, because they do not appear directly in the UBL data models but are logically implied by data type inheritance and do appear in the UBL XML schemas in accordance with the UBL Naming and Design Rules. As indicated here, a few of the most significant of these supplementary CCTS components become required XML attributes in UBL and will be required in any instance of an element derived from the corresponding type. See C.5 Navigating the UBL Data Model for an example of UBL attributes and a further discussion of this point. A reverse lookup of the implied occurrence of each attribute in the data models is provided in this summary report:

`mod/summary/reports/All-UBL-2.2-Documents.html#UDT`

UBL Unqualified Data Type	Definition	Attributes
AmountType	A number of monetary units specified using a given unit of currency.	**currencyID (required)** currencyCodeListVersionID
BinaryObjectType	A set of finite-length sequences of binary octets.	format **mimeCode (required)** encodingCode characterSetCode uri

(continued)

UBL Unqualified Data Type	Definition	Attributes
		filename
GraphicType	A diagram, graph, mathematical curve, or similar representation.	*not used in UBL 2.2*
PictureType	A diagram, graph, mathematical curve, or similar representation.	*not used in UBL 2.2*
SoundType	An audio representation.	*not used in UBL 2.2*
VideoType	A video representation.	*not used in UBL 2.2*
CodeType	A character string (letters, figures, or symbols) that for brevity and/or language independence may be used to represent or replace a definitive value or text of an attribute, together with relevant supplementary information.	listID listAgencyID listAgencyName listName listVersionID name languageID listURI listSchemeURI
DateTimeType	A particular point in the progression of time, together with relevant supplementary information.	*format (not used in UBL 2.2)*
DateType	One calendar day according the Gregorian calendar.	
TimeType	An instance of time that occurs every day.	
IdentifierType	A character string to identify and uniquely distinguish one instance of an object in an identification scheme from all other objects in the same scheme, together with relevant supplementary information.	schemeID schemeName schemeAgencyID schemeAgencyName schemeVersionID schemeDataURI schemeURI
IndicatorType	A list of two mutually exclusive Boolean values that express the only possible states of a property.	format
MeasureType	A numeric value determined by measuring an object using a specified unit of measure.	**unitCode (required)** unitCodeListVersionID
NumericType	Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.	format
ValueType	Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.	*not used in UBL 2.2*
PercentType	Numeric information that is assigned or is determined by calculation, counting, or sequencing and is expressed as a percentage. It does not require a unit of quantity or unit of measure.	format
RateType	A numeric expression of a rate that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.	format
QuantityType	A counted number of non-monetary units, possibly including a fractional part.	unitCode unitCodeListID unitCodeListAgencyID unitCodeListAgencyName

(continued)

UBL Unqualified Data Type	Definition	Attributes
TextType	A character string (i.e. a finite set of characters), generally in the form of words of a language.	languageID languageLocaleID
NameType	A character string that constitutes the distinctive designation of a person, place, thing, or concept.	languageID languageLocaleID

Some UBL BBIEs have data type qualifications based on the unqualified UBL types. These qualified types are all code types, and their definitions are the mechanism whereby a specific set of values is associated with each code.

UBL data type qualifications are expressed formally in an OASIS [16] (Context/Value Association) file contained in the `cva` directory of the 2.2 distribution.

`cva/UBL-DefaultDTQ-2.2.cva`

The specification of the CVA mechanism and format is maintained by the OASIS Code List Representation Technical Committee.

A human-readable version is provided in an accompanying HTML file, which also serves as primary documentation on the UBL codes defined as qualified data types.

`cva/UBL-DefaultDTQ-2.2.html`

The `val` directory contains the predefined CVA associations compiled into an XSLT file, `UBL-DefaultDTQ-2.2.xsl`, which is used in the recommended two-phase validation process to perform a check of code list values. See Annex E UBL 2.2 Code Lists and Two-phase Validation for a description of this process.

`val/UBL-DefaultDTQ-2.2.xsl`

The UBL revised approach to data type qualification contrasted to the UBL 2.0 approach is illustrated in the following diagram.

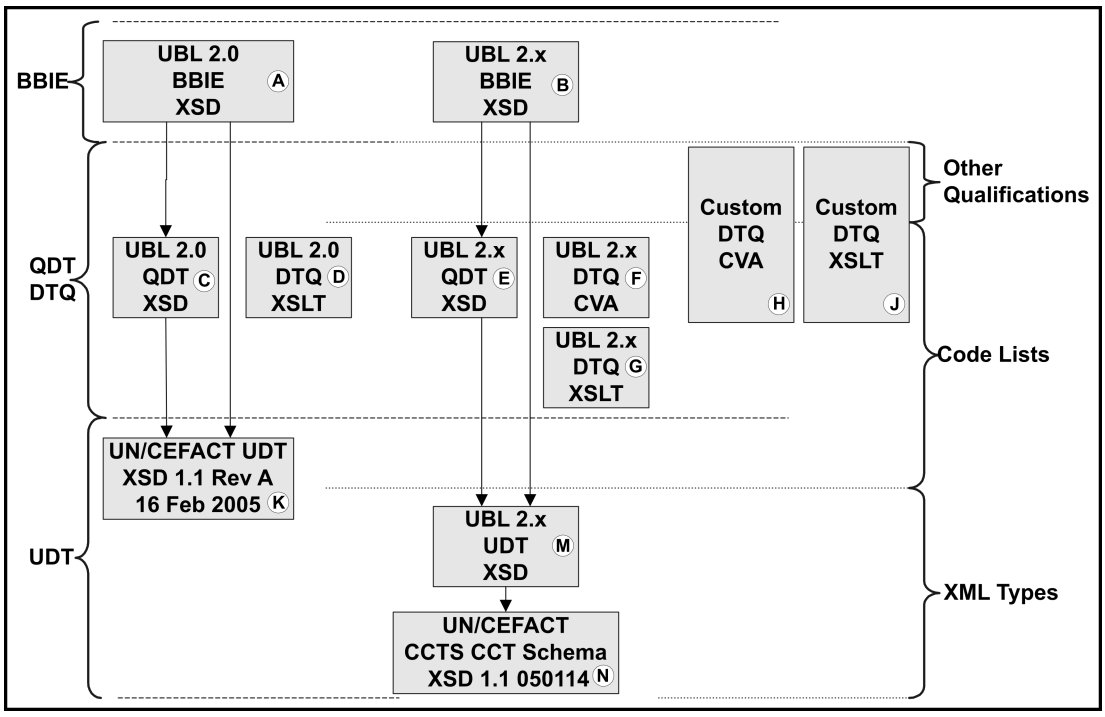

Figure D.1 — Data Type Qualification in UBL

In UBL 2.0, the schema library of common basic components (basic business information entities or BBIEs, **(A)** in the diagram) is based on a combination of the data types defined in the file of UBL 2.0 qualified data types **(C)** and the unqualified data types defined in the UN/CEFACT Unqualified Data Type schema module Ver. 1.1 Rev A 16 Feb 2005 **(K)**. The UBL 2.0 data type qualifications XSLT stylesheet **(D)** was used in the two-pass validation process, offering limitations on values such as code lists hardwired in the UN/CEFACT UDT definition.

In subsequent releases of UBL, the schema library of common basic components (**(B)** in the diagram) is based on a combination of the data types defined in the file of UBL qualified data types **(E)** and the data types defined in a file of UBL unqualified data types **(M)**. The latter inherits the data type definitions in the UN/CEFACT CCTS CCT schema module Ver. 1.1 050114 **(N)**. The UBL data type qualifications CVA file **(F)** controls the creation of the UBL XSLT stylesheet **(G)** used in the two-pass validation process, offering both limitations and extensions to values such as code lists. While this XSLT file, `UBL-2.x-DefaultDTQ.xsl`, can, when modified, apply to data type qualifications in general (such as field length restrictions and value range restrictions), the version of this file included in the UBL release contains only code list values linked to the metadata of the applicable code list.

The two remaining boxes on the right in the diagram illustrate that users can add further data type qualifications if desired by preparing a custom CVA **(H)** and creating a custom XSLT file **(J)** to replace the default CVA and XSLT stylesheet provided in the UBL distribution.

Users intending to prepare a custom CVA should note that `cva/UBL-DefaultDTQ-2.2.cva` contains relative URIs that expect the UBL 2.0 code lists from the UBL 2.0 Update Package in a sibling directory named `os-UBL-2.0`, and the UBL 2.1 code lists from the UBL 2.1 distribution in a sibling directory named `os-UBL-2.1`. This is irrelevant to users of the pre-compiled `val/UBL-DefaultDTQ-2.2.xsl` file contained in the UBL package, but users wishing to create their own CVA file must first install the code lists of prior releases of UBL 2.0. To properly install the update, first download and install the original UBL 2.0 release:

`http://docs.oasis-open.org/ubl/os-UBL-2.0.zip`

Then download and install the UBL 2.0 update:

`http://docs.oasis-open.org/ubl/os-UBL-2.0-update-delta.zip`

Then download and install the UBL 2.1 release:

`http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.zip`

Complete installation instructions can be found in the each package. As indicated above, the `os-UBL-2.0/` and `os-UBL-2.1/` directories thus created must be siblings directories to the directory created by installing the UBL 2.2 package.

Annex E
(informative)

UBL 2.2 Code Lists and Two-phase Validation

E.1 Code Lists Introduction

Code lists—the sets of codes such as “FR” and “USD” that are used to specify countries, currencies, and so on—play an important role in UBL, just as they do in all electronic business messaging schemes. By default, UBL uses several lists of standard codes published by agencies such as ISO and UN/CEFACT, as well as various codes that are specific to UBL.

In UBL 1.0 (2004), standard and default code list values were enumerated directly in the UBL schemas. This allowed all UBL 1.0 instances to be validated in a single pass using generic XML XSD (W3C Schema) processors. However, the specification of the default values directly in the schemas also made it difficult to modify the code lists to suit individual trading partner relationships and impossible to extend the list of allowable code list values while still using the standard UBL schemas as published by OASIS.

To give users maximum flexibility in configuring and updating UBL code lists without changing the standard UBL schemas, UBL 2.0 introduced a two-phase validation model that has now been fully implemented in UBL 2.1 and beyond. In the first phase, the UBL instance is checked for structure and vocabulary against a standard UBL schema using a generic schema validator (or custom-built software performing the same function). This is exactly the same procedure used for validation in UBL 1.0, except that the schemas do not contain hardwired code list values. Then in an added second validation (or verification) phase, code list values in the instance are checked against values obtained from external code list configuration files using an XSLT 1.0 processor driven by an XSLT 1.0 stylesheet. The default code list values assumed by the UBL 2.2 specification are expressed as data type qualifications in a file named `UBL-2.2-DefaultDTQ.xsl` located in the `val` directory, as described in more detail below. Publicly available tools were used to create the XSL file using the methodology described in the “Validation” section of [15], the UBL Guidelines for Customization.

Separating the checking of structure and vocabulary from the checking of code values allows trading partners to easily and precisely specify code list subsets and extensions and to apply them not just to individual UBL document types but also to particular elements and sub-trees within UBL document instances. Another way to say this is that the UBL code list methodology allows different versions of the same code list to be used in different document contexts. Thus, for example, a business in Canada might agree with a business in the United States to use a set of code list configuration files that allow the Buyer to be associated with either a U.S. state or a Canadian province but restrict the Seller to just U.S. states—that is, to apply a code list subset containing state and province codes in one place in a document instance and a different code list subset containing just state codes in another place in the instance.

E.2 Default Validation Setup

To facilitate the processing of UBL instances using the two-phase method, an “out-of-the-box” collection of open-source software that can be used to demonstrate default validation of UBL documents is included in the `val` directory of this release package. The validation harness assumes a Linux or Windows system with no currently installed XML or XSLT processing software.

The Java Runtime Environment (JRE) 1.5 or later is required to use the programs in the `val` directory; JRE versions below 1.5 will throw an error from the `xjparse.jar` module used to invoke the Xerxes schema parser. If necessary, download and install the latest JRE from the following location before continuing:

`http://www.java.com/en/download/manual.jsp`

To demonstrate UBL default validation:

1) Change to the `val` directory.

2) From within that directory, enter the test command

   `test.bat` (Windows)

or

`sh test.sh` (Linux)

The output, which is explained in the next section, should resemble the output shown in the following figure (the spacing has been manually adjusted to make the output easier to read).

```
############################################################
Validating order-test-good.xml
############################################################
============== Phase 1: XSD schema validation ==============
No schema validation errors.
============ Phase 2: XSLT code list validation ============
No code list validation errors.
############################################################
Validating order-test-bad1.xml
############################################################
============== Phase 1: XSD schema validation ==============
Attempting validating, namespace-aware parse
Error:file:///c:/d/ubl/2/val/order-test-bad1.xml:48:23:cvc-complex-type.2.4.a:
Invalid content was found starting with element 'cbc:ChannelCod'.
One of '{"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":ChannelCode,
"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":Channel,
"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":Value}' is expected.
Parse succeeded (0.822) with 1 error and no warnings.
############################################################
Validating order-test-bad2.xml
############################################################
============== Phase 1: XSD schema validation ==============
No schema validation errors.
============ Phase 2: XSLT code list validation ============
Value supplied ' LA ' is unacceptable for codes identified by 'ChannelCodeType'
in the context: cbc:ChannelCode
Processing terminated by xsl:message at line 18
```

Figure E.1 — Validation test output

3) From within the `val` directory, you can now validate any UBL document against the UBL schemas by executing commands of the form

   `validate<ubl-schema> <ubl-document>`

   where *<ubl-document>* is the path of a document to be validated and *<ubl-schema>* is the path of the UBL schema for that document type (Order, Invoice, etc.). For example, the scripts `val/testsamples.bat` and `val/testsamples.sh` show this process being used to validate the sample XML instances in the `xml` directory.

E.3 Discussion of the Default Validation Test

The test output displayed above demonstrates the default validation process with three test files: a valid UBL Order (`val/order-test-good.xml`); a UBL Order containing a bad (misspelled) element (`val/order-test-bad1.xml`); and a UBL Order that is schema-valid but contains an illegal code list value (`val/order-test-bad2.xml`). The file `val/test.bat` (Windows) or `val/test.sh` (Linux) is used to run the script `val/validate.bat` or `val/validate.sh` against each of the test files.

The first run using `order-test-good.xml` demonstrates both phases of the default validation process running normally. In the first phase, a standard W3C Schema (XSD) validator, Xerxes, is invoked from `val/w3cschema.bat` (or `val/w3cschema.sh`) to validate the specified UBL document (`.xml`) against the specified UBL runtime schema (`.xsd`). Since the input is a valid UBL Order, the output of the first phase simply indicates that the file is valid against the given Order schema.

The second phase of validation uses a standard XSLT 1.0 engine, Saxon, to verify that the values of various codes used in the UBL document to be tested (currency codes, packaging types, etc.) are valid in terms of the default UBL code list values specified in `val/UBL-DefaultDTQ-2.2.xsl`. Here the output line “No code list validation errors” from the `validate`

script indicates that the Saxon run (invoked from `val/xslt.bat` or `val/xslt.sh`) finds no illegal code values in the document.

The second run shows what happens when the input document (`order-test-bad1.xml`) contains an actual structure or vocabulary error, in this case due to omission of the trailing "e" from the element named `cbc:ChannelCode`. When the Xerxes parser encounters the malformed element name, it emits the error message shown in the example, and the `validate` script reacts to a non-zero status code from `w3cschema.bat` (or `w3cschema.sh`) by terminating the validation process.

In the third run, the input document `order-test-bad2.xml` is structurally valid according to the Order schema, but it contains an illegal code list value (the ChannelCode "AL" for cell phone has been mistyped as "LA"). Thus it passes the first phase when tested against the schema but fails the second phase when tested against `val/UBL-DefaultDTQ-2.2.xsl`.

To summarize, input documents are checked in the first validation phase for correctness of structure and vocabulary, using the constraints expressed in the appropriate UBL schema, and then they are checked in the second phase for correctness of default code list values, using the default constraints expressed in the XSLT file `UBL-DefaultDTQ-2.2.xsl`. This process is illustrated in the following diagram.

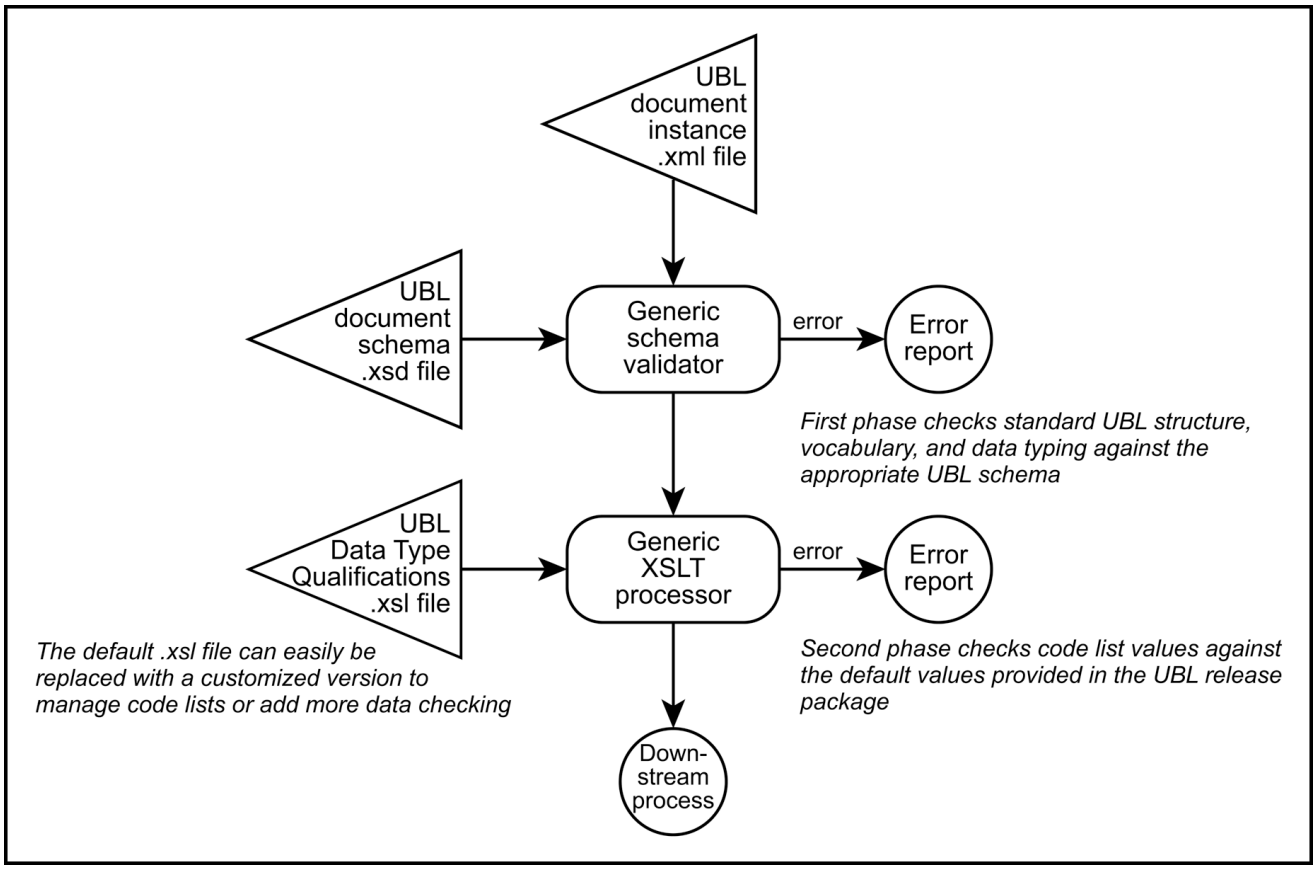

Figure E.2 — Two-phase Default UBL 2.2 Validation

It should be clear from the foregoing that the second phase of the default validation process can safely be omitted if it is considered unnecessary to check code list values. However, the reverse is not true; the second phase depends for correct operation on a prior check for structural validity, and therefore it will not give reliable results if run in the absence of the first (schema) validation phase.

E.4 Customizing the Default XSLT File

The validation framework provided in the `val` directory can be used to implement code list changes, define variant code lists to fit specific trading partner agreements, or associate different versions of the same code list with different parts of the same UBL document by substituting a custom process (be it XSLT or some other language or process) for the default `UBL-DefaultDTQ-2.2.xsl` provided in the UBL 2.2 distribution. This allows extensive code list management without the need to change the standard UBL 2.2 schemas. Schematron-based [27] techniques for generating a custom XSLT file to take the

place of UBL-DefaultDTQ-2.2.xsl are explained in [16] and [15]. See also Annex D Data Type Qualifications in UBL for more about UBL data type qualifications.

Since XSLT is a very powerful general-purpose XML transformation tool, the same framework can be extended to perform fairly sophisticated business rule checking by manually coding additional logic into the XSLT file that drives the second validation phase. Such modification is beyond the scope of the customization methodologies associated specifically with UBL, but a business analyst willing to perform XSLT programming can use this mechanism to offload a large proportion of input filtering from the back-end business application to a simpler input processing area. Additional XSLT scripts can be added to extract logical sub-trees of incoming UBL documents for allocation to different downstream processes and to perform even more extensive front-end processing.

E.5 Sources for the Default Validation Framework

Components of several freely available software distributions were used to create the val directory. Sources are given below so that these components can be updated as later releases become available.

— The file val/xjparse.jar (renamed from xjparse-2.0.1.jar) and the files in the "val/lib" directory are from the Xjparse 2.0.1 distribution at

http://xjparse.org

— The file val/saxon.jar is from the Saxon 6.5.5 distribution at

http://prdownloads.sourceforge.net/saxon/saxon6-5-5.zip

— The file val/UBL-DefaultDTQ-2.2.xsl was created using the Schematron [27] implementation of CVA files for validation at

http://www.CraneSoftwrights.com/resources/ubl/#cva2sch

E.6 Code Lists Included in UBL 2.2

E.6.1 Code List Format

The code lists included in the UBL 2.2 distribution use an OASIS Standard XML format for code lists called [20]. Each code list in the distribution is expressed as a genericode file. The code lists of UBL 2.0, UBL 2.1 and UBL 2.2 are incorporated into the default validation framework. Documentation on the UBL code lists is contained in a generated report file:

cva/UBL-DefaultDTQ-2.2.html

The code list files in UBL 2.2 are divided into two subdirectories, cl/gc/default and cl/gc/special-purpose.

E.6.2 cl/gc/default

The code lists in the cl/gc/default directory contain the default code values represented in UBL-DefaultDTQ-2.2.xsl. A second-phase code list check using an unmodified version of the test setup from this distribution as described above will verify all occurrences of code values from these lists against the values specified in the cl/gc/default directory. These are the code lists expected to be used in most application contexts, but there is no obligation to use them. The genericode files with corresponding "including deprecated" or "including deleted" files have been culled of deprecated or deleted values in order to be used in typical contexts. The files with entries no longer used are included for completeness.

cl/gc/default/AllowanceChargeReasonCode-2.2.gc
cl/gc/default/BinaryObjectMimeCode-2.2.gc
cl/gc/default/BinaryObjectMimeCode-2.2-incl-deprecated.gc
cl/gc/default/ChannelCode-2.2.gc
cl/gc/default/CountryIdentificationCode-2.2.gc
cl/gc/default/CurrencyCode-2.2.gc
cl/gc/default/LanguageCode-2.2.gc

cl/gc/default/PackagingTypeCode-2.2.gc
cl/gc/default/PackagingTypeCode-2.2-incl-deleted.gc
cl/gc/default/PaymentMeansCode-2.2.gc
cl/gc/default/TransportEquipmentTypeCode-2.2.gc
cl/gc/default/TransportModeCode-2.2.gc
cl/gc/default/UnitOfMeasureCode-2.2.gc
cl/gc/default/UnitOfMeasureCode-2.2-incl-deleted.gc
cl/gc/default/WeighingMethodCode-2.2.gc

Annex F
(informative)

UBL 2.2 Example Document Instances

The `xml` directory of this distribution contains a number of sample UBL documents that can be used for testing purposes. The `testsamples.bat` batch file and the `testsamples.sh` script in the `val` directory of this distribution can be used to demonstrate the validity of these examples in Windows and Linux operating environments. See Annex E UBL 2.2 Code Lists and Two-phase Validation for a general discussion of UBL validation methodology. For convenience, those examples that relate specifically to a particular document type are linked from the description of that type in 6.2 UBL 2.2 Document Schemas.

Example instances containing extensions

xml/MyTransportationStatus.xml
xml/UBL-Invoice-2.0-Enveloped.xml

Example instances related to signatures (see 8.5 Digital Signature Examples)

xml/UBL-Invoice-2.0-Detached-Signature.xml
xml/UBL-Invoice-2.0-Detached.xml
xml/UBL-Invoice-2.0-Enveloped.xml

Example instances with unconventional use of namespace bindings

xml/UBL-Invoice-2.0-Example-NS1.xml
xml/UBL-Invoice-2.0-Example-NS2.xml
xml/UBL-Invoice-2.0-Example-NS3.xml
xml/UBL-Invoice-2.0-Example-NS4.xml

Example instances of different versions of certain document types

xml/UBL-BusinessCard-2.2-Example.xml
xml/UBL-CreditNote-2.0-Example.xml
xml/UBL-CreditNote-2.1-Example.xml
xml/UBL-DebitNote-2.1-Example.xml
xml/UBL-DespatchAdvice-2.0-Example.xml
xml/UBL-DigitalAgreement-2.2-Example.xml
xml/UBL-DigitalAgreement-2.2-Example-Multilateral.xml
xml/UBL-DigitalCapability-2.2-Example.xml
xml/UBL-ExceptionCriteria-2.1-Example.xml
xml/UBL-ExceptionNotification-2.1-Example.xml
xml/UBL-ExpressionOfInterestRequest-2.2-Example.xml
xml/UBL-Forecast-2.1-Example.xml
xml/UBL-ForecastRevision-2.1-Example.xml
xml/UBL-ForwardingInstructions-2.0-Example-International.xml
xml/UBL-FreightInvoice-2.1-Example.xml
xml/UBL-FulfilmentCancellation-2.1-Example.xml
xml/UBL-GoodsItemItinerary-2.1-Example.xml
xml/UBL-InstructionForReturns-2.1-Example.xml
xml/UBL-InventoryReport-2.1-Example.xml

xml/UBL-Invoice-2.0-Example.xml
xml/UBL-Invoice-2.1-Example.xml
xml/UBL-Invoice-2.1-Example-Trivial.xml
xml/UBL-Order-2.0-Example.xml
xml/UBL-Order-2.0-Example-International.xml
xml/UBL-Order-2.1-Example.xml
xml/UBL-OrderCancellation-2.1-Example.xml
xml/UBL-OrderChange-2.1-Example.xml
xml/UBL-OrderResponse-2.1-Example.xml
xml/UBL-OrderResponseSimple-2.0-Example.xml
xml/UBL-OrderResponseSimple-2.1-Example.xml
xml/UBL-PriorInformationNotice-2.2-Example-Embedded.xml
xml/UBL-PriorInformationNotice-2.2-Example-External.xml
xml/UBL-ProductActivity-2.1-Example-1.xml
xml/UBL-ProductActivity-2.1-Example-2.xml
xml/UBL-ProductActivity-2.1-Example-3.xml
xml/UBL-Quotation-2.0-Example.xml
xml/UBL-Quotation-2.1-Example.xml
xml/UBL-ReceiptAdvice-2.0-Example.xml
xml/UBL-Reminder-2.1-Example.xml
xml/UBL-RemittanceAdvice-2.0-Example.xml
xml/UBL-RequestForQuotation-2.0-Example.xml
xml/UBL-RequestForQuotation-2.1-Example.xml
xml/UBL-RetailEvent-2.1-Example.xml
xml/UBL-SelfBilledCreditNote-2.1-Example.xml
xml/UBL-Statement-2.0-Example.xml
xml/UBL-StockAvailabilityReport-2.1-Example.xml
xml/UBL-TradeItemLocationProfile-2.1-Example.xml
xml/UBL-TransportationStatus-2.1-Example.xml
xml/UBL-TransportationStatusRequest-2.1-Example.xml
xml/UBL-TransportExecutionPlan-2.1-Example.xml
xml/UBL-TransportExecutionPlanRequest-2.1-Example.xml
xml/UBL-TransportProgressStatus-2.1-Example.xml
xml/UBL-TransportProgressStatusRequest-2.1-Example.xml
xml/UBL-TransportServiceDescription-2.1-Example.xml
xml/UBL-TransportServiceDescriptionRequest-2.1-Example.xml
xml/UBL-Waybill-2.0-Example-International.xml
xml/UBL-WeightStatement-2.2-Example.xml

Annex G
(informative)

Alternative Representations of the UBL 2.2 Schemas

UBL 2.2 continues the practice, adopted at the beginning of the UBL effort, of creating its normative XML specifications using W3C Schema (XSD) syntax. Alternative representations of the same content are technically non-normative, but are generated directly from the XSD and, with the exception of the UBL 2.2 digital signature extension (see 8.4 UBL Extension for Enveloped XML Digital Signatures), are intended to implement the same document instance constraints.

Regarding creating RELAX-NG [25] expressions of the UBL document models, the free Trang tool found at https://github.com/relaxng/jing-trang is suitable for converting the UBL W3C Schema expressions into such expressions.

Annex H
(informative)

The Open-edi reference model perspective of UBL

ISO/IEC 14662:2010 Information technology - Open-edi reference model [23] has been developed primarily in order to provide standards required for the inter-working of organizations through interconnected information technology systems. Open-edi lowers barriers to electronic data interchange by introducing standard business scenarios and the necessary services to support them.

The Open-edi Reference Model identifies the required standards for Open-edi and provides a reference for those standards by defining the basic concepts used to develop them.

Figure H.1 depicts two views to describe the relevant aspects of business transactions:

— the Business Operational View (BOV);

— the Functional Service View (FSV).

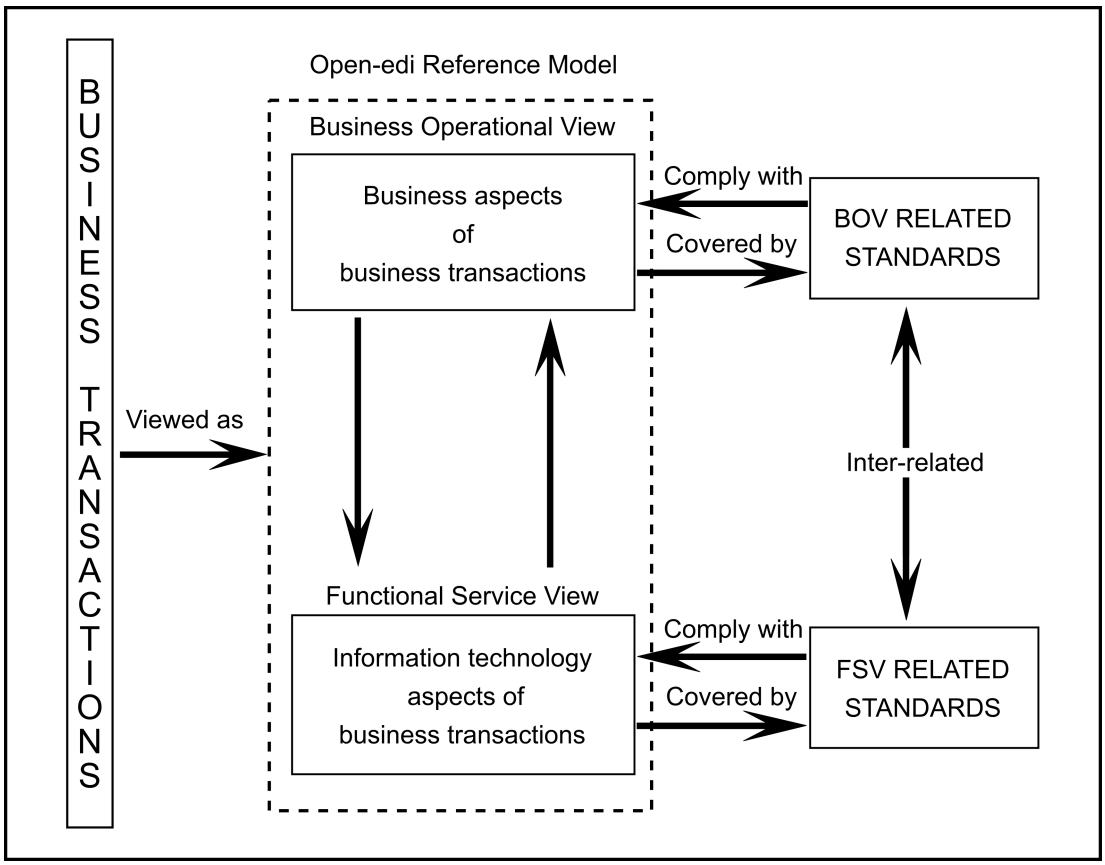

Figure H.1 — Open-edi Overview

The BOV addresses the aspects of the semantics of business data in business transactions and associated data interchanges which apply to the business needs of Open-edi. The BOV-related standards are tools and rules by which users who understand the operating aspects of a business domain may create scenarios.

The FSV addresses the supporting services meeting the mechanistic needs of Open-edi, focusing on information technology aspects of functional capabilities, service interfaces, and protocols.

Using the concepts of Open-edi, UBL provides a generic Open-edi Configuration that an Open-edi Community may customize with their own requirements to implement their own Open-edi Configuration.

ISO/IEC 15944-20 Information technology - Business operational view - Linking business operational view to functional service view [11] presents the relationships linking the BOV with the FSV.

Figure H.2 illustrates how the two normative deliverables of UBL, the semantic components and the XML schemas, align respectively with the BOV and FSV views of the Open-edi Reference Model.

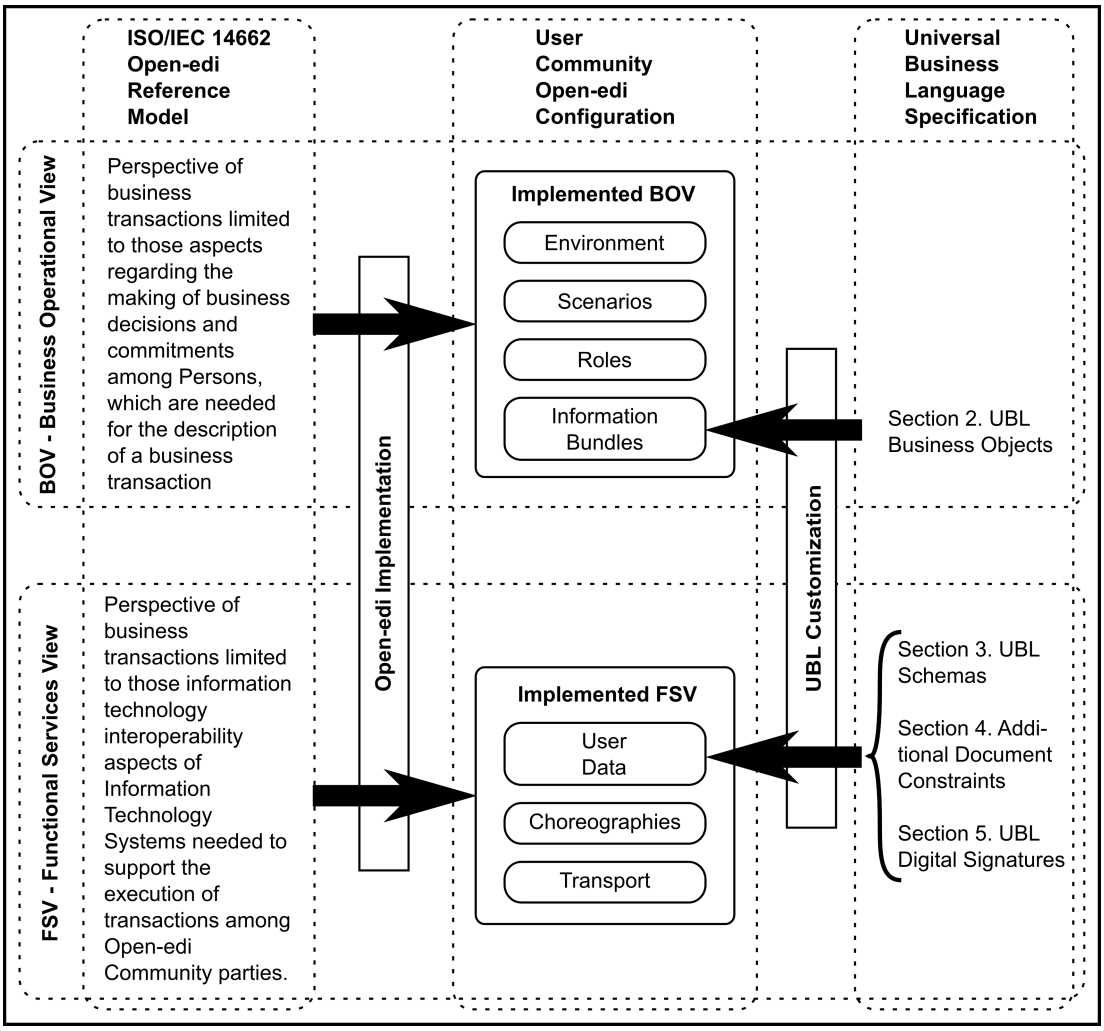

Figure H.2 — Open-edi Application

5 UBL 2.2 Business Objects provides the configuration's BOV with a suite of normative business objects and associated semantics from which the community selects the semantic components needed in an information bundle. An information bundle describes the semantics of the recorded information to be exchanged between Open-edi Support Infrastructures servicing Decision Making Applications. The community's configuration combines these information bundles with their identified scenarios and roles.

6 UBL 2.2 Schemas and 7 Additional Document Constraints provides the configuration's FSV with a set of corresponding normative XML schemas and document instance rules constraining the expression of the business objects in user data. One translates the semantic component values into a transfer syntax from the information bundle specification as a set of recorded information. It is the UBL XML syntax for the sets of recorded information defined by the information bundles that is exchanged between Parties.

8 UBL Digital Signatures provides the configuration's FSV with a normative schema fragment suitable for including profiles of advanced digital signatures in user data.

The other aspects of the implemented BOV and implemented FSV of the community's Open-edi Configuration are governed by influences outside of the scope of UBL. Those aspects guide the community in customizing UBL to suit their requirements, as outlined in A.4 UBL Customization.

Annex I
(informative)

Acknowledgements

The OASIS UBL Technical Committee thanks Altova for its contribution of XML Spy licenses for use in UBL schema design; Sparx Systems for its contribution of Enterprise Architect licenses for use in developing UML content models and swim-lane diagrams; SyncroSoft for its contribution of oXygen licenses used in DocBook authoring of UBL documentation; RenderX for its contribution of XEP licenses used in generating PDF documents from DocBook originals; and Crane Softwrights for the generation of the summary reports.

The following persons and companies participated as members of the OASIS UBL Technical Committee during the four years of its development (2013–2017).

Todd Albers, Federal Reserve Bank of Minneapolis
Oriol Bausà Peris, Individual
Kenneth Bengtsson, Alfa1lab
Erlend Klakegg Bergheim, Difi-Agency for Public Management and eGov
Peter Borresen, Document Engineering Services Limited
Jon Bosak, Individual
Roberto Cisternino, Individual
Robin Cover, OASIS
Kees Duvekot, RFS Holland Holding B.V.
Martin Forsberg, Swedish Association of Local Authorities & Regions
G. Ken Holman, Crane Softwrights Ltd.
Ole Ellerbæk Madsen, DIGST Denmark
Tim McGrath, Document Engineering Services Limited
Andrew Schoka, Individual
Enric Staromiejski Torregrosa, everis, S.L.U.
Audun Vennesland, SINTEF

Annex J
(informative)

Temporary Annex - Rendering per ISO/IEC Directives, Part 2

NOTE This temporary appendix will be removed in the final version of the committee specification.

During the review process of UBL 2.2, the distribution includes the parallel production of the specification PDF for publishing as ISO/IEC 19845 using the page layout prescribed by “ISO/IEC Directives, Part 2”.

This rendering is found at ISO-IEC-19845.pdf. This rendering is not included in the final distribution but its content is submitted directly to ITTF for publishing.

Annex K
(informative)

Temporary Annex - Change Log

NOTE This temporary appendix will be removed in the final version of the committee specification.

Revision	Date	Editor	Changes made
csd02wd 02	04 October 2017	GKH	Prepare dated version of CSD02WD02 for voting
csd02wd 02	30 September 2017	GKH	Incorporate CSD02 changes (models and code lists)
csprd01	18 January 2017	GKH	Cover page changes requested by TC Admin
csprd01	21 December 2016	GKH	Copy of CSD01 with changed cover page
csd01	21 December 2016	GKH	Updated with known spelling mistakes, new status
csd01wd 13	11 December 2016	GKH	Revised tendering diagram, renamed tendering documents, updated tickets
csd01wd 12	28 November 2016	GKH	Revised hub diagrams. Changed model names. Updated tickets.
csd01wd 11	18 November 2016	GKH	Revised hub text; revised digital agreement models.
csd01wd 10	05 November 2016	GKH	Release of latest draft schemas and new hub document edits.
csd01wd 09	30 October 2016	GKH	Release of latest draft schemas and new document types; revised table of contents (still missing some content)
csd01wd 08	30 September 2016	GKH	Release of latest draft schemas and new document types
csd01wd 07	12 September 2016	GKH	Release of latest draft schemas
csd01wd 06	08 September 2016	GKH	Changes to accommodate simultaneous publishing for both OASIS and ISO/IEC ITTF requirements
csd01wd 02	27 June 2016	GKH	Initial version of 2.2 with few integrity checks and only some structural changes

Bibliography

7 Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

8 Commission Decision 2011/130/UE of the European Commission of 25 February 2011 on establishing minimum requirements for the cross-border processing of documents signed electronically by competent authorities under Directive 2006/123/EC of the European Parliament and of the Council on services in the internal market.

9 Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC). ETSI TS 102 918 V1.1.1, April 2011.

10 Business Document Naming and Design Rules Version 1.0. Edited by Tim McGrath, Andy Schoka and G. Ken Holman. 14 July 2016. OASIS Committee Specification 01. http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.0/cs01/Business-Document-NDR-v1.0-cs01.html. Latest version: http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.0/Business-Document-NDR-v1.0.html.

11 ISO/IEC 15944-20 Information technology - Business operational view - Linking business operational view to functional service view

12 UN/CEFACT Core Component Technical Specification, Version 2.01

13 Voluntary Interindustry Commerce Standards, Collaborative Planning, Forecasting, and Replenishment Version 2.0, Global Commerce Initiative Recommended Guidelines, June 2002

14 CPFR: An Overview, 18 May 2004

15 OASIS Committee Specification 01, UBL 2 Guidelines for Customization, First Edition, 25 December 2009

16 OASIS Context/value association using Genericode 1.0. 15 April 2010. Committee Specification 01 http://docs.oasis-open.org/codelist/ContextValueAssociation/doc/context-value-association.html.

17 CEN Workshop Agreement: E-invoices and digital signatures (CWA 15579), July 2006.

18 CEN Workshop Agreement: Storage of Electronic Documents (CWA 15580), July 2006.

19 Reference Architecture of eBusiness in Textile Clothing and Footwear Sector

20 OASIS Code List Representation (Genericode) Version 1.0. 28 December 2007. Committee Specification 01. http://docs.oasis-open.org/codelist/approved/genericode/oasis-code-list-representation-genericode.html.

21 UBL Maintenance Governance Procedures Version 2.2. Edited by Ole Madsen, Tim McGrath and G. Ken Holman. 04 March 2015. OASIS Committee Note 01. http://docs.oasis-open.org/ubl/UBL-Governance/v1.0/cn01/UBL-Governance-v1.0-cn01.html. Latest version: http://docs.oasis-open.org/ubl/UBL-Governance/v1.0/UBL-Governance-v1.0.html.

22 ISO/IEC 11179-1:1999 Information technology — Specification and standardization of data elements — Part 1: Framework for the specification and standardization of data elements

23 ISO/IEC 14662:2010 Information technology - Open-edi reference model

24 OASIS Standard, Open Document Format for Office Applications (OpenDocument) Version 1.2 - Part 3 Packages, December 2006.

25 ISO/IEC 19757-2, Information technology — Document Schema Definition Language (DSDL) — Part 2: Regular-grammar-based validation — RELAX NG , Information technology — Document Schema Definition Language (DSDL) — Part 2: Regular-grammar-based validation — RELAX NG AMENDMENT 1: Compact Syntax

26 Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), August 2001.

27 Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation (Schematron)

28 UBL Naming and Design Rules Version 3.0. Edited by G. Ken Holman. 20 July 2016. OASIS Committee Note 01. http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/cn01/UBL-NDR-v3.0-cn01.html. Latest version: http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/UBL-NDR-v3.0.html.

29 Unified Modeling Language Version 1.5 (formal/03-03-01)

30 ISO 14533-2:2012 Processes, data elements and documents in commerce, industry and administration -- Long term signature profiles -- Part 2: Long term signature profiles for XML Advanced Electronic Signatures (XAdES)

31 John Boyer, Canonical XML Version 1.0, 15 March 2001.

32 Anders Berglund, et al., XML Path Language (XPath) Version 2.0, 23 January 2007.

33 Steven DeRose, et al., XML Pointer Language (XPointer) Version 1.0 Working Draft, 16 August 2002.

34 XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999

35 Michael Kay, XSL Transformations (XSLT) Version 2.0, 2007-01-23.

ICS

Price based on 157 pages