Universal Business Language 1.0 Beta – Committee Draft

17 November 2003

Document identifier:

UBLTC-Library-beta-1.0-cd-03

Location:

http://www.oasis-open.org/committees/ubl/lcsc/UBLv1-beta

Editors:

Bill Meadows, Sun Microsystems <bill.meadows@sun.com>
Lisa Seaburg, Aeon LLC <lseaburg@aeon-llc.com>

Contributors:

Members of the Technical Committee

Abstract:

This specification defines the Library for the Universal Business Language.

Status:

This document is a Committee Draft of the OASIS Universal Business Language (UBL) Technical Committee. The OASIS UBL Technical Committee invites interested parties to comment on this release directly to the UBL Library Content Subcommittee Editor, Bill Meadows.

Table of Contents

1 Introduction

Since its introduction 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 huge waste 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 backoffice 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 the interoperability problem by defining a generic XML interchange format for business documents that can be extended to meet the requirements of particular industries. Specifically, UBL provides the following:

- A library of XML schemas for reusable data components such as "Address," "Item," and "Payment" -- the common data elements of everyday business documents.
- A small set of XML schemas for common business documents such as "Order," "Despatch Advice," and "Invoice" that can be used in a generic order-to-invoice trading context.
- Guidelines for the extension of UBL in specific trading relationships.

A standard basis for XML business schemas is expected to have 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 small and medium-size enterprises (SMEs).
- Standardized training, resulting in many skilled workers.
- A universally available pool of system integrators.

The adoption of UBL is also expected to foster the creation of inexpensive data input and output tools and to provide a universally understood and recognized commercial syntax for legally binding business documents.

The design of UBL schemas is modular, reusable, and extensible in XML-aware ways. The analysis and design processes used by the UBL Library Content team are described in Section 3.0 Library and Methodology. The UBL Library has been designed as a collection of object classes, their properties and associations expressed as a conceptual model. We call these components Business Information Entities (BIES). These Business Information Entities (BIES) are assembled into a specific hierarchical, document models, such as an Order or an Invoice. These document models are then transformed based upon specific UBL Naming and Design Rules [NDR] into XML Schema syntax [XSD1][XSD2].

By publishing the models, methodology and rules for schema creation, we hope that UBL components will also be used to assemble new and customised document structures. UBL is designed to be layered on existing successful standards. For example, the ebXML infrastructure developed by OASIS and the UN/CEFACT provides for XML registry services, reliable XML messaging, standardized trading partner agreements, a standard data registry, and a business process methodology.

UBL also provides an XML implementation of Electronic Business XML (ebXML) Core ComponentsTechnical Specification (v2.0).

Significantly, UBL leverages knowledge from existing EDI and XML B2B systems. It is user-driven, with deep experience and partnership resources to call on. Our goal is to unite and harmonize a number of currently existing XML and EDI business libraries into a set of legally recognized international standards.

UBL is committed to truly global trade and information interoperability. UBL will be freely available to everyone without legal encumbrance or licensing fees.

To aid in deployment, the normative standard UBL schemas are accompanied by a multitude of non-normative supporting materials, some of which are included in this package and some of which are available from referenced sites. These materials include:

- UML class diagrams of the conceptual models on which the schemas are based;
- UML class diagrams describing the documents themselves;
- descriptions of two example implementations;
- sample instances of each of the UBL documents used in those implementations;
- formatting specifications for sample renderings of those instances; and
- an ASN.1 specification to enable the transmission of UBL messages in binary form.

1.1 Notes about this Release

This release, known as UBL 1.0 Beta Committee Draft, is provided to enable trial implementations of UBL in realistic business environments. It is not an OASIS Technical Specification. There are certain features we would like to bring to the attention of implementors.

1.1.1. Recursive structures

Certain components in the library participate in a nesting that may result in recursion. For example, a Package may contain other Packages, a Delivery may specify another Delivery, etc. This is a legitimate business construct. In any implementation these would be constrained by some degree of limitation to the depth of recursion. We cannot describe this constraint in the schema. Therefore, it is theoretically possible to create unbounded document instances where these structures are used. Implementors should be aware of this and may wish to guard against this in their applications.

1.1.2. Implementation of Core Components Technical Specification

The UBL Library does not currently define any UBL-specific Data Types, as specified in the Core Component Technical Specification [CCTS]. The only DataTypes used in this release are the Data Types of primary and secondary Representation Terms.

1.1.3 Code Sets

The method for validating against enumerated code lists described in this document has not been fully implemented in UBL 1.0 Beta. This work is under review by the UBL Code List Subcommittee but is not expected to impact document instances created with the current schemas.

1.2 Scope

The Library Content part of UBL specifies a library of business information entities to be used in the construction of business documents together with a set of common XML business documents assembled from those entities.

This normative sections of this document are:

- the context scenario and business rules used to construct the business models and business documents;
- a W3C Schema (XSD) of re-usable components;
- the W3C Schemas (XSD) of the business documents required for the context scenario.

1.3 Support for this Release

The downloadable version of this release is available from UBLv10-beta Downloadable Release. (This is a zip file that will unpack to give you a replica of the online release directories.)

If there are any problems with the links in this document, you can find the full online version at:

http://www.oasis-open.org/committees/ubl/lcsc/UBLv10-beta/ .

On release of this Committee Draft, a publicly subscribable mail list will be created for the discussion of UBL among software developers. Archives of this mail list will be found at

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

In addition UBL has established a Pilot and Implementation Subcommittee to assist trial implementors in their application of this specification.

Once in operation, subscriptions to both lists can be made through the OASIS list manager at:

http://lists.oasis-open.org/ob/adm.pl

1.4 The OASIS UBL TC

The work of the OASIS UBL Technical Committee and its various Subcommittees is open to public view through the mail archives linked from the UBL home page: http://www.oasis-open.org/committees/ubl

1.5 Document Conventions

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 [RFC2119] as quoted here:

> MUST: This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute requirement of the specification.
>
> MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.
>
> SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
>
> SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
>
> MAY: This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to inter-operate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to inter-operate with another implementation which does not include the option (except, of course, for the feature the option provides).

1.6 Disclaimer

This document and its associated components are Copyright © 2003 OASIS and are protected by applicable law as works in progress within the OASIS Universal Business Language Technical Committee. As works in progress, they do not yet have the status of an OASIS Standard or an OASIS Committee Specification. This draft and its associated components are provided on a royalty-free basis and may be freely circulated for purposes of experimentation and review. While the construction of experimental prototypes based on these materials is encouraged for the purpose of generating input back to the committee process, implementers are strongly advised against basing commercial or mission-critical applications on the draft specifications contained in this package. THESE MATERIALS ARE FURNISHED WITH NO WARRANTY, EXPRESS OR IMPLIED, AS TO THEIR SUITABILITY FOR ANY APPLICATION.

2 Context of Initial Library [NORMATIVE]

2.1 Initial UBL Business Scenario

The specific context adopted for UBL 1.0 is based on a typical trading cycle that of procurement. We have used this context as a means of developing a set of common, re-usable Business Information Entities and their accompanying document definitions.

This section describes the scenario, business rules, transactions and choreography of a rudimentary order-to-invoice business process. A set of UBL documents have been assembled to demonstrate the information exchanges required by these transactions. We have adopted an 80/20 rule for this scenario - recognising this is not the definitive description of this process but a generalised case.

Of course, this is not the entire scope of the UBL Library. The components and their documents can also be used as a basis for extension to create more function-rich, but separately defined, scenarios. As this occurs, we envisage that this section will become part of a registry of available business processes from different, complementary sources.

2.2 The Order-to-Invoice Business Process

This model addresses the requirements of a basic, usable trading cycle from Order to Invoice between Buyer and Seller. It includes specifications for:

- Order
- OrderChange
- Order Response (simple)
- Order Response (complex)
- Order Cancellation
- Despatch Advice
- Receipt Advice
- Invoice

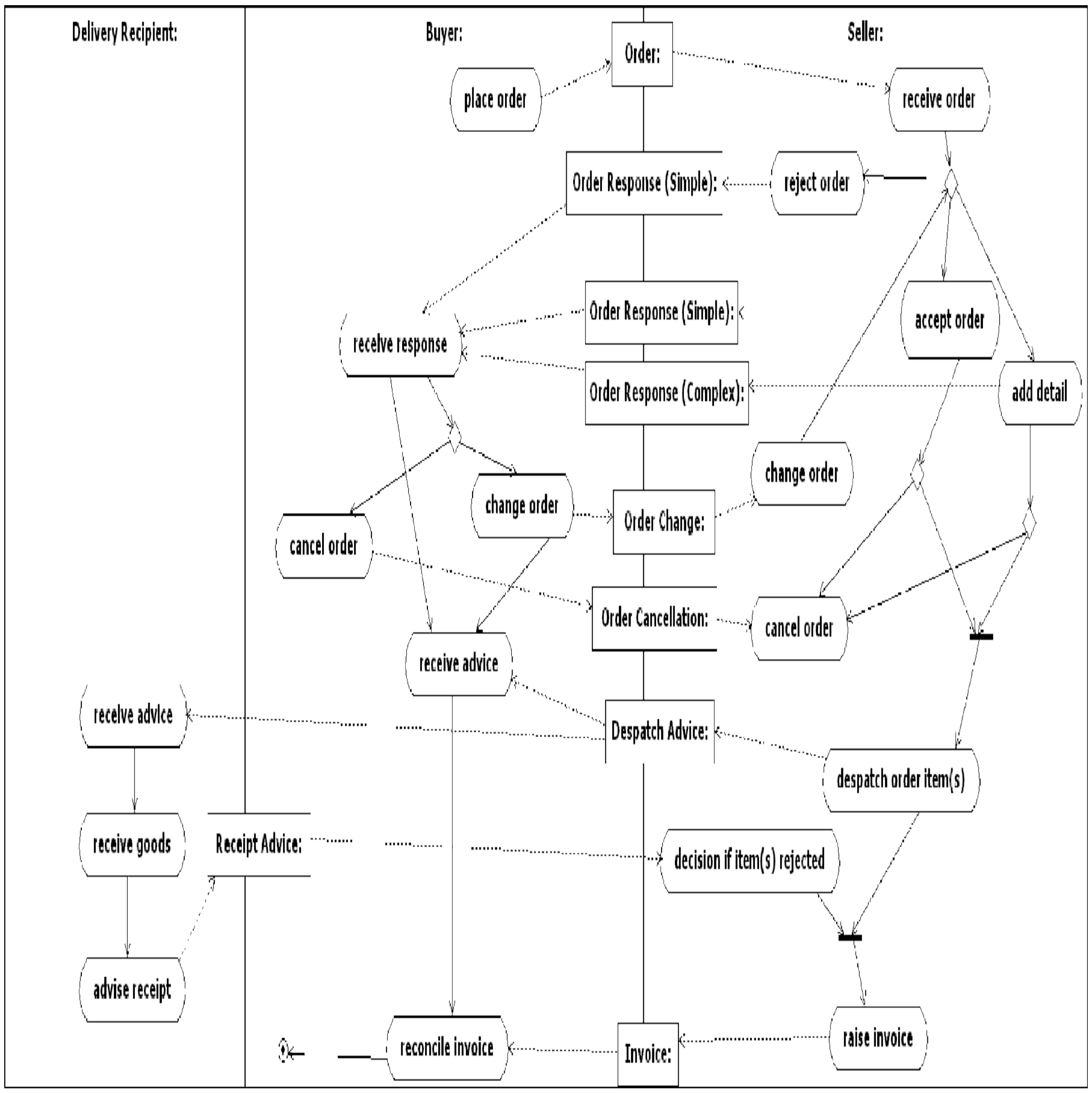


Figure 1. Order-to-Invoice Business Process

Items

An Identifier identifies each Item (e.g. a product identifier), which shall be one of the following:

- Buyer's Item Identification, or
- Seller's Item Identification, or
- Manufacturer's Item Identification, or
- Catalogue Item Identification, or
- Item Identification according to a Standard body's system.

The Item Identification assumes that each different packaging of an Item (e.g. a 6-pack and a 12-pack of the same item) has a different Item Identifier.

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, product code and where it is necessary to provide additional descriptive information about the item to precisely identify what is required.

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 Measurements

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

Other Item Details

For an Item, price ranges by amount, quantity, etc. are not repeated back to the Seller; only the active price is specified. The Buyer may not know the Item Base Price, in which case it is not specified. This makes a detailed response from the Seller necessary *[See Order Response (Complex)]*.

Ordered items may include Hazardous items, insofar as it is not necessary to specify related information 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.

Order

The Order may specify Charge Payment (e.g. freight, documentation etc) instructions that identify the type of charge and who pays which charges. The Order can be placed 'on account' against a trading credit account held by the Seller, or against a credit/debit card account, or a direct debit agreement. The Order overall allows only for specification of Currency (e.g. £, $, € etc by ISO currency code) for Pricing, for Invoice presentation, for Tax accounting. In the case of International freight/documentation charges, it may also be necessary to specify the Currency.

Trade discount may be specified at 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 Order Response (Complex)]*.

The Order may specify delivery terms and constraints that apply for the delivery location in relation to the following information that would normally not appear until the Despatch Advice:

- Transport
    - Means
    - Mode
    - One- to many-legged journey

        - Dates
        - Locations
    - Arrival 'window'
- Consignment packaging
    - Type, e.g. Container, Pallet
    - Identifier, e.g. SSCC, Shipping label (Despatch Advice)

The Order provides for multiple Order Lines.

Order Lines

Each Order Line provides for specification of a single place of delivery, and a schedule of quantities and requested delivery dates.

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

The Buyer may indicate potential alternatives that are acceptable. For each Order Line, an Alternative Item can be included. The Alternative Item may be specified by any one of the range of Item identifiers. For example, the specified Quantity may change e.g. 20x6-packs as an alternative to 10x12-packs.

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 fulfill without change or that the Order has been rejected.

Order Response (Complex)

Proposed changes by the Seller would be accomplished through the OrderResponse (Complex).

The Order Response (complex) is a complete replacement of the Order. It reflects the entire state of the 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
- Trade Discount
- Charges
- Customs Commodity Classification codes

The Seller may advise replacements or substitutes which will be made, or changes necessary, using the Order Response (complex). The Substitute or Replacement Item may be specified by any one of the range of Item identifiers. For example, the specified Quantity may change e.g. 20x6-packs as a replacement for 10x12-packs.

Order Change

The Buyer can change an Order, subject to the legal contract or trading partner agreement, by sending an OrderChange, or by sending an Order Cancellation followed by a new, complete replacement, Order.

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

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

Order Cancellation

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

Despatch Advice

The following information may appear in the Despatch Advice:

- Transport
    - Means
    - Mode
    - One- to many-legged journey
        - Dates
        - Locations
    - Arrival 'window'
- Consignment packaging
    - Type, e.g. Container, Pallet
    - Identifier, e.g. SSCC, Shipping label (Despatch Advice)

The Despatch Advice caters for two situations:

- Organisation of the delivery set of items by Transport Handling Unit(s) so that the Receiver can check Transport Handling Unit and then 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.
- Organisation 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 can 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 may not correspond one-to-one with Order Lines, but these need to be linked by reference. The information structure of the Despatch Advice, geared to physical considerations, 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.

Receipt Advice

The Receipt Advice is sent by the Receiver (Buyer) to the Seller to confirm receipt of items, and is capable of reporting shortages and/or damaged items.

The Receipt Advice caters for two situations. For ease of processing claimed receipt against claimed delivery, it needs to be organised in the same way as the matching Despatch Advice:

- 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.
- 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 Receiver to state any shortages from the claimed despatch quantity, to state any quantities rejected for a given reason.

As presently arranged the Receipt Line only allows for one rejection quantity and reason. However, any rejection of quantities of same item for different reasons could be achieved by subdividing the Receipt Line so that there are multiple Receipt Lines to one Despatch Line.

Invoice

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:

- Pre-payment invoice (payment expected)
- Pro-forma invoice (pre 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 re-iterate information already established in the Order, Order Change, Order Response (complex), Despatch Advice, or Receipt Advice that is not necessary when invoicing. The Invoice refers to the Order, Despatch Advice or Receipt Advice by a Reference of those documents.

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

Charges can 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

The present Invoice does not cover Debit and Credit Notes. Nor does the cycle include a Customer Account Statement that summarises Invoices, Credit Notes and Debit Notes to be paid.

Invoice Item Line

Each Invoice Line refers to the related Order Line and may refer to the Despatch Advice Line and/or Receipt Advice Line.

Adapting UBL for other scenarios

Different business scenarios to meet different ways of trading cycle operation can, and should, be developed by separate, appropriate business experts. Ideally they should take advantage of the basic UBL model as a starting point and as an exemplar. However, part of the UBL charter is to develop a methodology which will formalize the way that documents for other scenarios can be implemented. This is known as UBL Context Methodology [CM]. When this is in place as part of UBL 2.0 it will promote greater interoperability, reduce ambiguity, and avoid unnecessary overlap.

Meanwhile we encourage the UBL community to share their customisation and developments, both to improve the quality of the underlying library and provide valuable input into the UBL customisation methodology.

For example, within the procurement domain, suggested other scenarios include situations of:

- Vendor managed inventory
- Self-billing
- Master Order and Call-offs
- Prior Quote Request & Quotation
- International Trade requiring Multi-party Transportation
- Hire Trade (e.g. tool hire, scaffolding hire), etc.

3 Library and Methodology [NON-NORMATIVE]

It is not the purpose here to give a tutorial on the development process nor is the intention to define in detail the way UBL has used various tools and techniques. The sole normative deliverable of UBL is the schemas: unlike some other standards initiatives UBL does not mandate the use of a specific formal development method.

However, a development methodology has evolved during the UBL project. We refer to this approach as Document Engineering.

The purpose of this section is to describe the process that evolved, so that users can understand better the role of the various technical artifacts developed by UBL, and the tools that are available to work with these artifacts.

The initial library of business information entities (BIEs) was based upon the xCBL3.0 schema library. After a review of these it was felt necessary to create an abstracted model of the entities in a syntax neutral form which would support better an iterative development lifecycle. This abstraction is known as the UBL conceptual model. This modelling language used is UML.

It is important to understand that the conceptual model was developed as a means to an end. The end result is the UBL schemas and the UBL schemas are the sole normative artifacts of the UBL development process. At present there is no automated process that takes the conceptual model and generates the input to the next stage in the development process - currently this is the spreadsheet of BIEs. However, the conceptual model will be maintained by UBL and it is this model that will be used by UBL as the starting point for any modifications to the UBL.

The next stage of the process was to identify and document the artifacts required by the ebXML Core Component Technical Specification (CCTS) - Aggregate Business Information Entities (ABIE)s, their Basic Business Information Entities (BBIE) properties and their Associations with other ABIEs (ASBIE)s. This was a manual process using business knowledge of the domain, the UML diagrams, and the CCTS[CCTS]. The resultant BIEs were documented in a spreadsheet format. The reason for using a spreadsheet is that the conceptual model was not constructed with a UML profile that would facilitate the automated production of the XML schemas, and the development of and agreement to such a profile was seen as a potentially lengthy process. Conversely, it was a simple process to develop a spreadsheet format that would be both CCTS compliant and facilitate the automated production of schemas. It is the spreadsheet that is used to maintain the UBL Library. Importantly, it is spreadsheet that provides the additional meta-data and associated formulae to facilitate compliance with the CCTS.

Therefore, the BIEs identified in the model were transcribed manually into a spreadsheet of re-usable BIEs. Additional individual spreadsheets were developed for each document type in the initial UBL context scenario. These document models can be viewed as demonstrations of how UBL documents may be assembled.

This development process is shown in the diagram below.

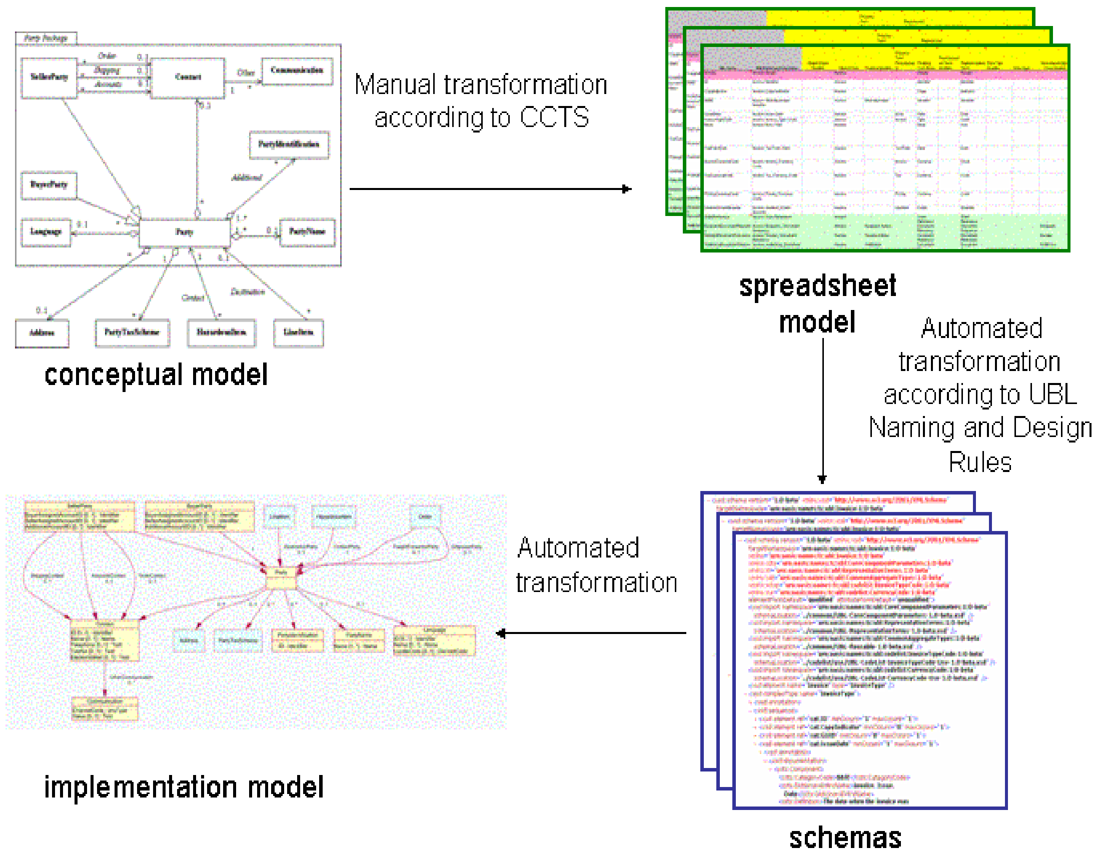

conceptual model
Manual transformation according to CCTS
spreadsheet model
Automated transformation according to UBL Naming and Design Rules
schemas
Automated transformation
implementation model

Figure 2. The Development Process

3.1 The Conceptual Model

The UBL conceptual model incorporates the data requirements of all of the documents supported by UBL 1.0. It was developed as a UML class diagram. The model is restricted to the data aspects of the UBL process scenario: it does not include other UML diagram notations such as use case models, interaction diagrams etc.

The conceptual model is the result of a detailed analysis of the data requirements to support the initial UBL Business Process Scenario. During the modeling process common items of data were identified by a process of normalization to identify aggregates based on functional dependency. Where appropriate these were generalized so that they could be re-used to support the various business documents.

The conceptual model is used for the following purposes:

- It facilitates the identification of the re-usable components - i.e. the data that are common across the business documents comprising UBL 1.0.
- It provides for the understanding of the total data scenario in a visual way
- It is the source from which the BIEs are derived and documented in a spreadsheet

The conceptual model is included in this document as a series of diagrams. For the purposes of clarity the model represented here does not include any attributes, nor does it contain any of the additional semantics that were developed to assist in the documentation of BIEs.

As an example, the Party re-usable component in UML is shown below.

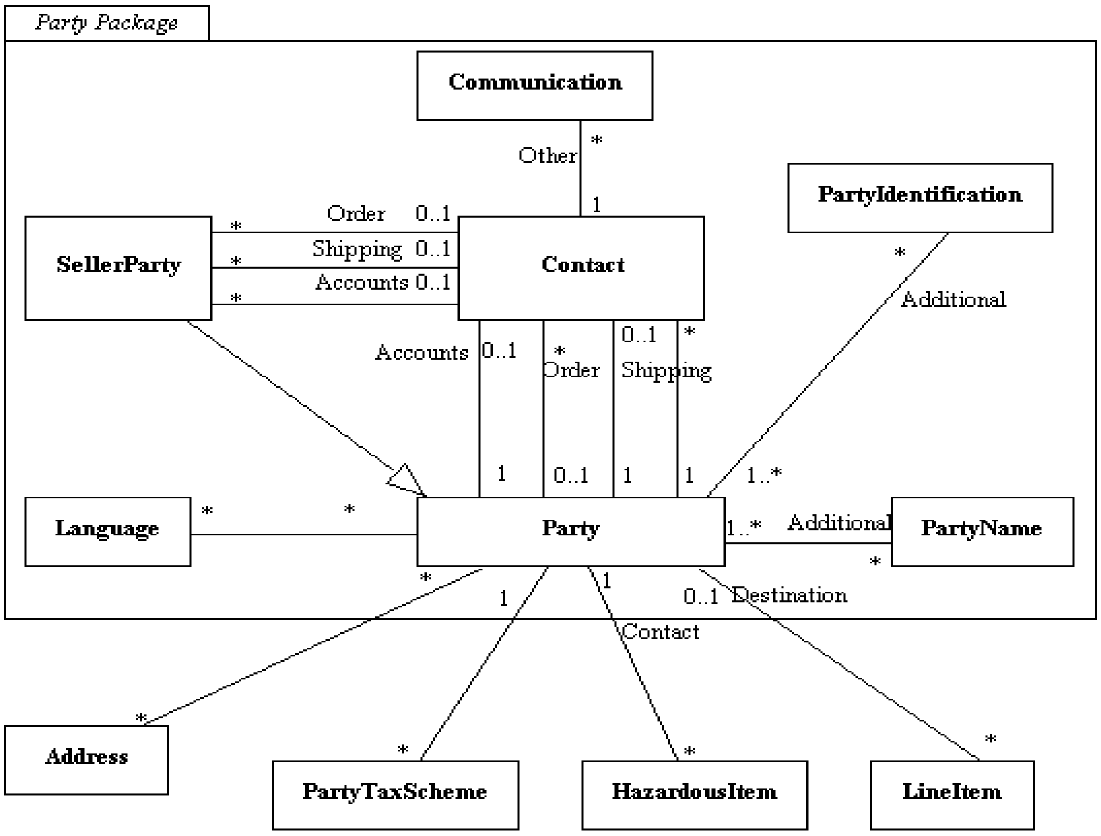


Figure 3. Conceptual UML class diagram of Party

The full list of class diagrams showing re-usable components in sets of packages is shown below.

Address

Contract

Delivery

Document reference

Hazardous item

Item

Party

Payment

Procurement

Tax

Each of the business documents comprising UBL 1.0 is documented as a class in the UML model. This class represents the top level Aggregate BIE for the document type. All the other BIEs for the business document were derived by traversing the associations from this class, and by applying knowledge of the hierarchy required. As an example, the conceptual model of the Order document is shown below.

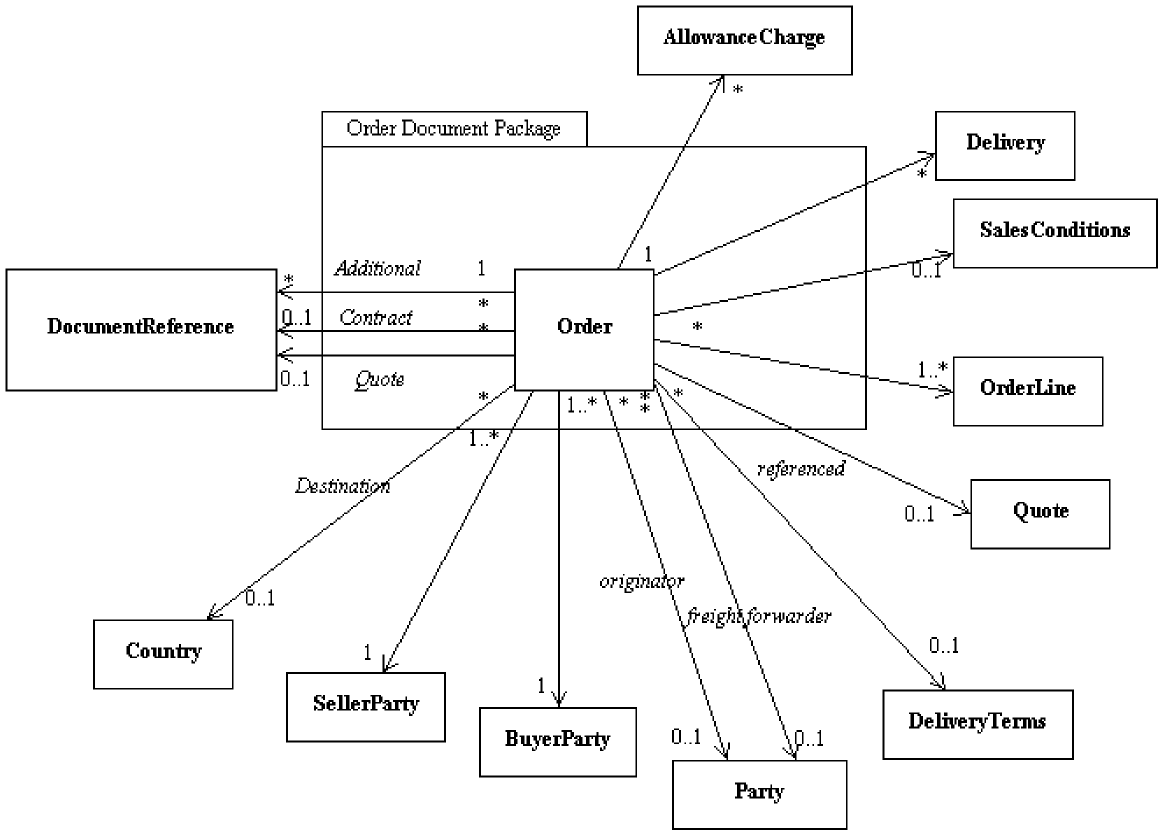


Figure 4. Conceptual UML class diagram of the Order Document

The full list of class diagrams for the business documents is shown below.

Order

Order response

Order change

Order cancellation

Despatch Advice

Receipt advice

Invoice

Outside of the internal UBL development process, this conceptual model is for information purposes only.

In addition to this, the model represented here is just a skeleton of the complete model (it contains only the classes and their associations). For these reasons the conceptual model is not a complete enough artifact for implementors to use if they wish to modify the UBL schemas to suit a specific business community.

3.2 Spreadsheet Models

The UBL team chose, at an early stage of development, to use spreadsheets as a working tool to maintain the document models. The library and its documents are composed of a combination of ABIEs, BBIEs and the relationships between two ABIEs, ASBIEs. Many of the spreadsheet columns are determined by requirements of the ebXML Core Components Technical Specification [CCTS], others by UBL Naming and Design Rules[NDR].

Each business information entity (BIE) is defined in a single row. Row background colour distinguishes between BBIE (white), ABIE (pink) and ASBIE (green). Annotations in the first row of each column provide further explanation of the conventions and design aspects of the spreadsheets.

All UBL document schemas are automatically generated from these spreadsheet models. Please note, that the normative form of UBL documents definitions is not the spreadsheet model but the XSD XML Schemas. The spreadsheets provide:

- - a suitable starting point for model editing and for Schema regeneration using a scripting or transformation tool such as that used by the UBL team.

  For those wishing to customise UBL or use it as the basis for a new vocabulary, the spreadsheets can be manually edited. It is intended that there be levels of conformance to UBL, depending on how customisation is performed. Any schema generation should be compliant with the UBL Naming and Design Rules [NDR] to promote compatibility of component libraries. Furthermore, UBL foresees the development of a customisation methodology for version 2.0 of the UBL..

  Modifying the current spreadsheets requires an understanding of their structure, the ebXML Core Components Technical Specification [CCTS] and the various UBL library constituents. For example, some columns are updated manually. Others have formulas in their cells which implement ebXML CCTS and UBL Naming and Design Rules [NDR]. Awareness of this is necessary when adding or editing the row contents. Care should be taken to avoid updating cells that contain formulae.

- - a supplementary, non-normative documentation of the UBL models

- - an aid to understanding the existing UBL architecture.

All Business Documents are defined in their individual spreadsheets, each references the Re-usable Component Library spreadsheet.

These are provided in both Microsoft(R) Excel (.xls) and Open Office formats (.sxc).

UBL Order (MS Excel) or UBL Order (Open Office)

UBL Order Response (Simple) (MS Excel) or UBL Order Response (Simple) (Open Office)

UBL Order Response (Complex) (MS Excel) or UBL Order Response (Complex) (Open Office)

UBL Order Change (MS Excel) or UBL Order Change (Open Office)

UBL Order Cancellation (MS Excel) or UBL Order Cancellation (Open Office)

UBL Despatch Advice (MS Excel) or UBL Despatch Advice (Open Office)

UBL Receipt Advice (MS Excel) or UBL Receipt Advice (Open Office)

UBL Invoice (MS Excel) or UBL Invoice (Open Office)

All Aggregate Business Information Entities are expressed in the UBL Re-usable Component Library spreadsheet (MS Excel) or UBL Re-usable Component Library spreadsheet (Open Office).

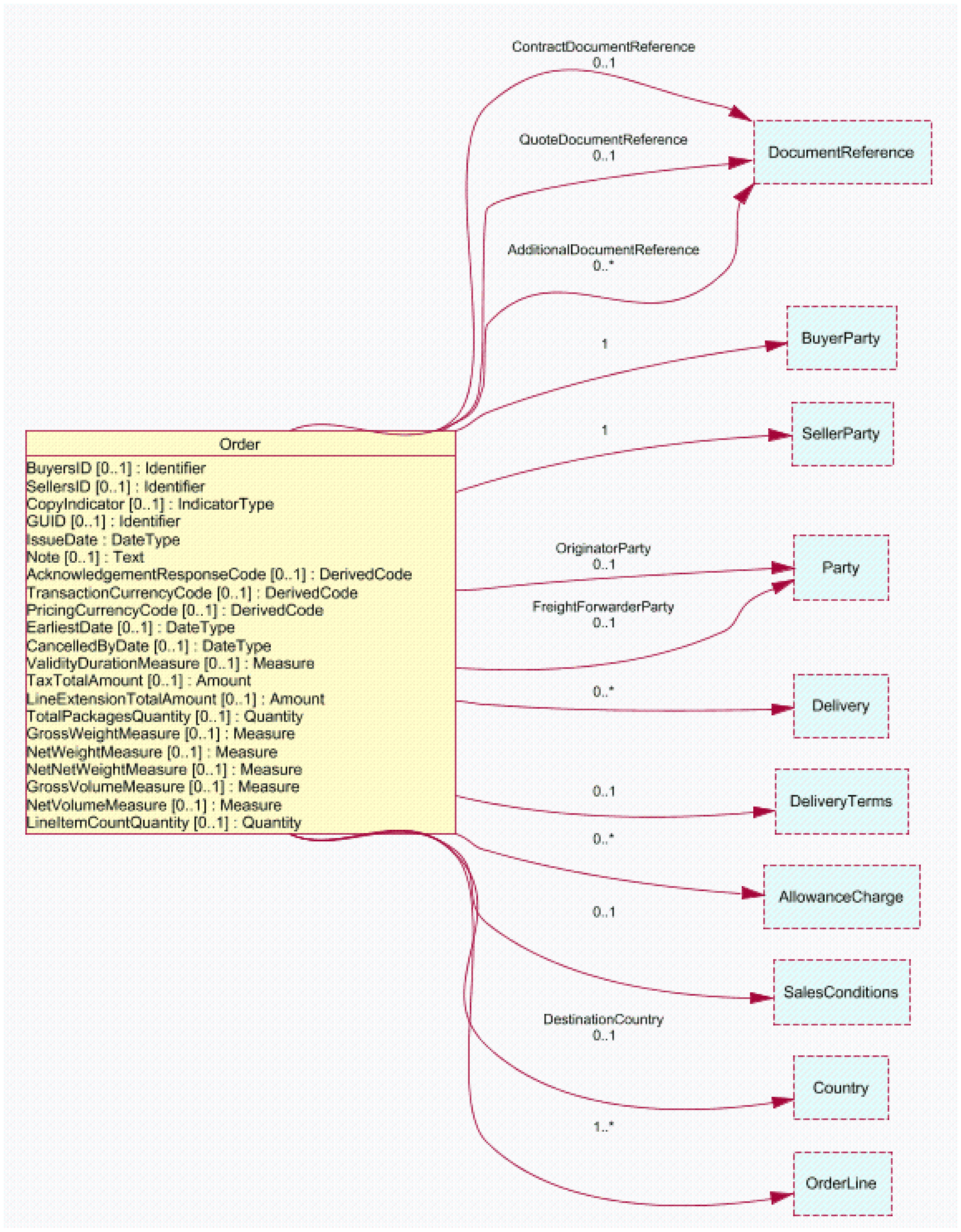
All Codelist information is expressed in the UBL-CodeListCatalogue-1.0-beta (MS Excel) or UBL-CodeListCatalogue spreadsheet (Open Office).

3.3 The Implementation Model

The implementation model of UBL represents the actual XML Schemas as a UML model. This is produced by automatically transforming the UBL XML Schemas into a model conformant with the Unified Modeling Language [UML]. This model is then used to produce a set of class diagrams that illustrate each of the main documents and several views of the reusable components. The automated transformation and diagram creation was performed using a Schema to UML transformation tools called Ontogenics' *hyper*Model.

These UML class diagrams are intended to assist understanding of the UBL Schemas, but without requiring that the reader understand the XML Schema syntax. The diagrams intentionally suppress some of the detail from the XML Schemas that is also represented in the reverse-engineered UML model. For example, this UML implementation model contains the sequence order of elements within a complex type definition, but this information is not included in the diagrams. Also, part of the transformation process from XML Schema to UML model is designed to create a useful object-oriented representation that could be used for other software engineering work based on this model (e.g. the OMG's model driven architecture). Consider two examples where this choice affects the resulting UML model. First, the "Type" suffix of XML Schema complexType names are removed when creating the UML class name to yield an object class name independent of XSD syntax. Second, complex type child elements with simple content values are represented in UML as class attributes, whereas elements with complex content are represented as associations to those type classes.

There are eight main business documents in the UBL 1.0 library and one class diagram is created for each of these document definitions. These document-level diagrams are presented as simplified views that suppress the detail of types contained within these aggregate structures. As an example, the class diagram for the UBL Order document is shown in this diagram:


ContractDocumentReference
0..1
QuoteDocumentReference
0..1
DocumentReference
AdditionalDocumentReference
0..*
1
BuyerParty
1
SellerParty
Order
BuyersID [0..1] : Identifier
SellersID [0..1] : Identifier
CopyIndicator [0..1] : IndicatorType
GUID [0..1] : Identifier
IssueDate : DateType
Note [0..1] : Text
AcknowledgementResponseCode [0..1] : DerivedCode
TransactionCurrencyCode [0..1] : DerivedCode
PricingCurrencyCode [0..1] : DerivedCode
EarliestDate [0..1] : DateType
CancelledByDate [0..1] : DateType
ValidityDurationMeasure [0..1] : Measure
TaxTotalAmount [0..1] : Amount
LineExtensionTotalAmount [0..1] : Amount
TotalPackagesQuantity [0..1] : Quantity
GrossWeightMeasure [0..1] : Measure
NetWeightMeasure [0..1] : Measure
NetNetWeightMeasure [0..1] : Measure
GrossVolumeMeasure [0..1] : Measure
NetVolumeMeasure [0..1] : Measure
LineItemCountQuantity [0..1] : Quantity
OriginatorParty
0..1
Party
FreightForwarderParty
0..1
0..*
Delivery
0..1
DeliveryTerms
0..*
AllowanceCharge
0..1
SalesConditions
DestinationCountry
0..1
Country
1..*
OrderLine

581 **Figure 5. Implementation Model for the Order Document**

582 In addition to the main document diagrams, there are ten class diagrams that present views of the
583 packages of reusable components used in these documents. For example, the Order diagram
584 includes associations to Party, SellerParty, and BuyerParty. The following figure illustrates the
585 detailed definitions of these components.

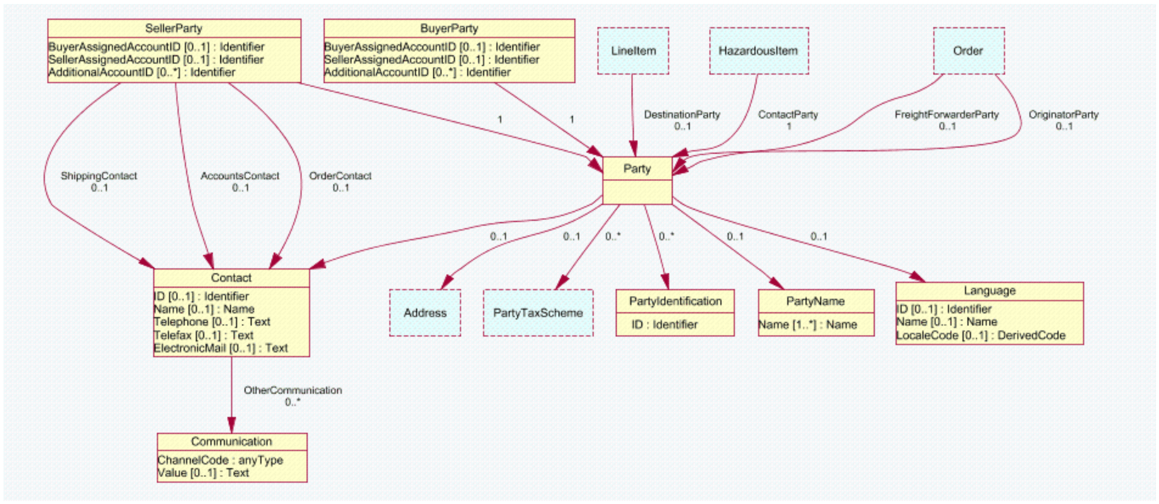


Figure 6. Implementation Model for Party Components

This implementation model was used by the UBL subcommittees to help verify the completeness and accuracy of the library definitions, but was not used to generate the XML Schemas contained in this specification. However, schema generation from UML models is theoretically possible and could be considered for extending or customizing the UBL library. Readers of this specification may find these diagrams helpful while gaining an understanding of the UBL library content and as a quick reference during future use of the schemas. In particular, business users who wish to review the library contents without learning the XML Schema language will find these model diagrams helpful.

The complete list of UML implementation model diagrams is:

Document Diagrams

- Order
- OrderCancellation
- OrderChange
- OrderResponse
- OrderResponseSimple
- Invoice
- DespatchAdvice
- ReceiptAdvice

Reusable Component Diagrams

- Address
- Contract
- Delivery
- DocumentReference
- HazardousItem
- Item
- Party
- Payment
- Procurement
- Tax

4 UBL Schemas [NORMATIVE]

The UBL Document Schemas form the essential deliverables of the UBL Technical Committee. The XML Schemas are implementations of the conceptual models identified by UBL, and are the only normative representation of the UBL library.

Within this release there are 3 main sub-directories under the “xsd/” directory: the “codelist/”, “common/”, and “maindoc/” sub-directories.

The sub-directories show the following contents:

Directory	Sub-directory	UBL edited schemas	Auto-generated schemas	Number of schemas
xsd/codelist/	etc/	-	1	1
	placebo/	-	56	56
	use/	-	56	56
xsd/common/		4	1	5
xsd/maindoc/		-	8	8

In the common directory, the 4 UBL edited schemas are:

UBL-CoreComponentParameters-1.0-beta.xsd
This file provides the structure description of fields that go into the annotation/documentation section of the type definitions used in all the other schemas. The meta information, such as the object class, representation terms, etc are stored in specific fields as defined in this CoreComponentParameters in a consistent format. This allows the source derivation information to be extracted instead of reverse-engineered or guessed.
UBL-CoreComponentTypes-1.0-beta.xsd
This file provides the Core Component Types (CCT) as defined by the UN/CEFACT Core Components Technical Specification team. The types defined within this file provide the basic building type blocks to construct higher level representation types in a standardized and consistent manner.
UBL-RepresentationTerms-1.0-beta.xsd
This file provides the Representation Terms (RT) that implements the basic type building blocks to construct main document schemas.
UBL-DataTypes-1.0-beta.xsd
This file is a placeholder to implement data types that are required by main document schemas, but which are currently not yet a CCT-recognized type yet. In this release of UBL, there is no such need for additional data types yet. The content of this schema is therefore empty, although the necessary namespace and imports are already set in place.

The only schema file in the 'common' sub-directory that is not manually crafted is the Reusable schema. This is automatically generated from the re-usable spreadsheet model.

UBL-Reusable-1.0-beta.xsd
This file provides the Aggregate Business Information Entities (BIEs) that are used throughout the UBL. Effectively, this schema serves as a “ABIE type-database” for constructing the main documents.

The “maindoc/” directory contains the 8 automatically generated schemas for each document type:

Directory	File Description	Purpose
xsd/maindoc/	UBL-DespatchAdvice-1.0-beta.xsd	This schema provides the UBL Despatch Advice document.
	UBL-Invoice-1.0-beta.xsd	This schema provides the UBL Invoice document.
	UBL-Order-1.0-beta.xsd	This schema provides the UBL Order document.
	UBL-OrderCancellation-1.0-beta.xsd	This schema provides the UBL Order Cancellation document.
	UBL-OrderChange-1.0-beta.xsd	This schema provides the UBL Order Change document.
	UBL-OrderResponse-1.0-beta.xsd	This schema provides the UBL Order Response document.
	UBL-OrderResponseSimple-1.0-beta.xsd	This schema provides the UBL Order Response Simple document.
	UBL-ReceiptAdvice-1.0-beta.xsd	This schema provides the UBL Receipt Advice document.

5 Code Lists

Editor's Note: the following description of a method for validating against enumerated code lists has not been fully implemented in UBL 1.0 Beta. This work is under review by the UBL Code List Subcommittee.

The primary objective of populating codes lists within the UBL Library is to promote interoperability. That is, by having known sets of values in enumerated lists we allow information to be exchanged unambiguously. We recognise that other information may be useful for presenting or describing these codes, but the most effective means of conveying this additional information is yet to be established. In UBL 1.0 we have concentrated solely on enabling interoperability by populating enumerated lists.

Strictly speaking a code is an abbreviation of a value. We recognize that in some cases the values in our lists are not codes but a controlled vocabulary of terms. However, the same mechanisms can be used to support both. This mechanism is what we refer to as the UBL code list architecture.

UBL has identified and detailed four validation perspectives, termed "code list definitions", for the values found in instance content of the type of a given code list, summarized as:

- Standard: These are mandatory codes that MUST be used to be UBL compliant. The reason a code is defined as standard may be that it required for correct use of business transactions (e.g. status codes), promotes a single, internationally recognised code set (e.g. currency code) or enforces a restricted set of possible values (e.g. latitude code).
  UBL will supply codes that should be sufficient to all users of UBL. The values used in instances should be validated against the supplied codes and validating processors should correctly throw errors when invalid values are used.
  The implementation of standard codes is as a "stock" code without a "placebo" (see below).
- Placebo: These are code lists whose values SHOULD be agreed upon between trading partners. UBL SHALL NOT enforce any validation of the coded values in these code lists. These are implemented by using the generic "normalized string" data type for these elements in which these coded values belong. Applications working with the instances have the responsibility of validating any content found for these codes.
- Stock: These are UBL-supplied sets of candidate codes available to be used in place of "placebo" code lists. Trading partners who agree to utilize the values supplied by UBL MAY choose to replace the "placebo" lists with these "stock" lists.
- Private-Use: Trading partners SHOULD always have the ability to create and then utilize sets of codes of their own choosing. "Private-use" code lists MAY replace either "standard" code lists or "placebo" code lists. Trading partners MAY choose to implement validation of private code lists either in the schema expression or in their applications but MUST do so without impacting on any other code list used.

All codes will be handled by separate schema modules, regardless of their source so that the necessary enumeration's and their subsequent maintenance will not impact the other library schemas.

There are two sources of codes for UBL code list definitions. The first is when the code list is created by an outside agency or organization (e.g. the UNCL TRED codes) and is available without fees or incumberances. The second is when no royalty-free external code list is available and UBL has created its own codes (e.g. OrderRejectionReasonCode). We envisage and encourage external code agencies to establish and maintain their own code schemas for use with UBL. However, in the first instance we accept that we will need to use localised UBL snapshots of the original codes, maintained by UBL. As external code list owners make their code lists available in the form of importable schema modules, the corresponding references for those code list modules can be changed accordingly.

Within the UBL schemas, an "in-use" directory is used to define each code list to be used during the validation process. Only values for standard definitions of code lists are validated for their content when UBL is run out-of-the-box. All other code lists are validated using the placebo definition merely as having a tokenized value, and this value is not checked against any further constraints. Customised implementations can chose to adopt either stock or private-use code list definitions, and after any such engagement can revert to the out-of-the-box configuration by engaging the original standard or placebo code list definition.

UBL provides a catalogue of the code lists in the UBL Library. This catalogue also describes other meta-data that may be of significance to users of the codes.

The “codelist/” directory contains 3 sub-directories:

Directory	Sub-directory	File Description	Purpose
xsd/codelist/	etc/	UBL-CodeListCatalogue-1.0-beta.xml	A master catalogue of all code lists that are used in one way or another within UBL schema deliverables. The catalogue also provides necessary meta data for the tool to generate consistent linkages between code list references, namespace values, filenames and other important aspects of code list schema generation.
	placebo/	-	
	use/	-	

The “placebo/” sub-directory contains a set of generated code list schemas that carry appropriate namespace values and prefixes so that the main documents could reference and import the code list schema type. In practical usage, however, the files in the “placebo/” sub-directory are not imported by any other schema; they are copied first into the “use/” sub-directory, and (with its filename) renamed from “*Placebo*.xsd” to “*Use*.xsd”. In this way, if and when an alternative implementation of code list schema is implemented by UBL in time to come, they could be copied and renamed in the “use/” sub-directory without upsetting any of the higher-level schemas that have used the previous code list schemas.

Following the current code list usage architecture, the schema files found in the “use/” sub-directory are therefore copies of exactly the same files found in the “placebo/” sub-directory. The idea is that if the code list schema in the “use/” sub-directory gets replaced by other code list schema implementation, it is possible to revert back by copying the corresponding code list schema found in the “placebo/” sub-directory.

Currently, a few alternative means of code list schema implementations are being examined within the UBL TC. The sub-directory structure may be expanded further in future. As the final structure of this directory is still being worked out, the current structure sets up in compatible preparation for this future expansion and change.

Annex F lists the files found in the “placebo/” and “use/” directory.

There is a large set of meta data associated with each of the code list schema. To get a sense of what each of the code list is intended for, how is it is being used, who is the authority, what is the version number, etc, one should look into the file “xsd/codelist/etc/UBL-CodeListCatalogue-1.0.xml”, where each <CodeListItem> child element within that file gives the set of meta data for that particular code list schema.

Appendix A. References

A.1 Normative References

[ISO11179] International Standards Organisation's Specification and Standardization of Data Elements for Information Technology

http://isotc.iso.ch/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm??Redirect=1

[ISO 8601] Data elements and interchange formats -- Information interchange -- Representation of dates and times

http://www.iso.org/iso/en/CombinedQueryResult.CombinedQueryResult?queryString=8601

[CCTS] UN/CEFACT ebXML Core Components Technical Specification 2.0

http://www.oasis-open.org/committees/download.php/4259/CEFACT%20CCTS%20Version%202%20of%2011%20August.pdf

[NDR] Universal Business Language Naming and Design Rules

http://www.oasis-open.org/committees/sc_home.php?wg_abbrev=ubl-ndrsc

[CM] Universal Business Language Context Methodology

wd-cmsc-cmguidelines-1.0-beta

[UML] Unified Modeling Language 1.3 (formal/02-07-01)

http://www.omg.org/cgi-bin/doc?formal/02-07-01

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

http://www.w3.org/TR/2000/REC-xml-20001006

[XSD1] XML Schema Part 1: Structures, W3C Recommendation 2 May 2001

http://www.w3.org/TR/xmlschema-1/

[XSD2] XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001

http://www.w3.org/TR/xmlschema-2/

A.2 Terms and Definitions

Business Context

The formal description of a specific business circumstance potentially identified by the values of a set of context categories, allowing different business circumstances to be uniquely distinguished.

Class Diagram

A graphical notation used by the UML [UML] to describe the static structure of a system, including object classes and their associations.

Container

A modular and self-contained group of data components.

Containership

Aggregating components (nested elements in an XML schema [XML]).

Context

The circumstance or events that form the environment within which something exists or takes place.

Dependency Diagram

A refinement of a class diagram that emphasis's the dependent associations to between object classes.

Document

A set of information components that are interchanged as part of a business transaction; for example placing an order.

Document Assembly

A description of an hierarchical pathway through a normalized model of information components.

Functional Dependency

A means of aggregating components base of whether the values of a set of properties change when another set of properties changes. That is whether the former is dependent on the latter.

Hierarchical Model

A tree-structured model that can be implemented as a document schema.

Normalization

A formal technique for identifying and defining functional dependencies.

Conceptual Model

A representation of normalized data components describing a potential network of relationships between aggregate components.

Schema

An XML document definition based on the W3C XML Schema language [XSD1][XSD2].

schema

Any XML document definition.

Spreadsheet Model

A representation of a data model in tabular form.

The terms *Core Component* and *Business Information Entity* are used in this specification with the meanings given in [CCTS].

The terms *Object Class, Property Term, Representation Term,* and *Qualifier* are used in this specification with the meanings given in [ISO11179].

A.3 Symbols and Abbreviations

ABIE

Aggregate Business Information Entity

ACC

Aggregate Core Component

ASBIE

Association Business Information Entity

ASCC

Association Core Component

BBIE

Basic Business Information Entity

BCC

Basic Core Component

BIE

Business Information Entity

CC

Core Component

EAN

European Article Numbering Association

EDI

Electronic Data Interchange

ISO

International Standards Organisation

NDR

UBL Naming and Design Rules [NDR]

UML

Unified Modeling Language [UML]

UN/CEFACT

United Nations Centre for Trade Facilitation and Electronic Business

XML

Extensible Markup Language [XML]

XSD

World Wide Web Consortium's XML Schema Language [XSD1][XSD2]

A.4 XML Naming and Design Rules

The complete UBL XML Naming and Design Rules (NDR) document is currently in active editing. It will be completed by and released with the final release of UBL.

The completed NDR document will be a fully annotated version of the rules checklist contained in the current release. Explanatory text is being developed around each rule to facilitate understanding and use of this rules document.

After the milestone meeting in Montreal, held July 28 through August 1, 2003, the NDR Sub Committee decided to give the Library Content Sub Committee a snapshot of the rules as they existed coming out of that meeting. It is this snapshot that this Beta Release is based on.

Highlights of these rules are:

- Adherence to the Core Component Technical Specification, 2.0, Dated August 2003.
- Implementation of the Core Component Types schema module.

This rules table reflects only those rules valid on 19 September 2003. The link to this table is: rn-ndrsc-v1-0-beta.html.

Appendix B. UBL Document Examples (Non-Normative)

B.1 Example One Buying Office Supplies

The buyer, Bill's Microdevices, orders several different items from an office supply store. He knows the supplier's codes for the items and the price.

Office Supply Order - XML instance, Office Supply Order - printed version

The buyer, decides to change the original order.

Office Supply Order Change - XML instance, Office Supply Order Change- printed version

The seller, Joe's Office Supply, replies with an Order Response (simple) so as to indicate the acceptance of the order. At the same time, the seller gives his reference number of the order, i.e. the sales order in his system, and also tells the buyer whom to contact if he has any queries.

Office Supply Order Response - XML instance (simple), Office Supply Order Response - printed version

The buyer cancels a different Order

Office Supply Order Cancel - XML instance, Office Supply Order Cancel - printed version

The seller advises the buyer of the despatch of the items ordered.

Office Supply Despatch Advice - XML Instance, Office Supply Despatch Advice - printed version

The buyer notifies the seller of missing items.

Office Supply Receipt Advice - XML Instance, Office Supply Receipt Advice - printed version

The Seller raises the Invoice automatically when the despatch occurs, and the resolution of shortages etc will be handled post-invoicing. The Invoice shows the tax amount The Seller notes that payment is due within 30 days of Invoice.

Office Supply Invoice - XML Instance, Office Supply Invoice - printed version

B.2 Example Two Buying Joinery

The buyer, Jerry Builders, PLC. in the UK, orders a number of windows, a door set and some lengths of timber for delivery to a building site. He knows the supplier's codes for the items and that he must also specify a number of physical attributes to get the precise item that he wants. Some windows are asymmetric and are 'handed' left or right: most door sets are handed as they are hinged on one side. The wood and its finish, the 'fittings' are the handles, stays etc. Items can be glazed in different ways. Loose timber is coded according to its cross section and the length must be specified. While the buyer knows these things from the catalogue he does not know the current prices or any discount rate he may get.

Joinery Order - XML Instance, Joinery Order - printed version

The seller, Specialist Windows PLC, replies with an Order Response (complex) so as to indicate the unit price of each item and to inform the buyer of the trade discount that he will be given. At the same time, the seller gives his reference number of the order, i.e. the identity of the order in his system, and also tells the buyer whom to contact if he has any queries.

Joinery Order Response - XML Instance, Joinery Order Response - printed version

The seller advises the buyer of the despatch of the items ordered, which will in fact be delivered on two pallets identified as "A" and "B" (i.e. transportation units). The Despatch Advice lists the items in order line sequence and refers to the pallet on which the item is delivered.

Joinery Despatch Advice - XML Instance, Joinery Despatch Advice - printed version

The Despatch Advice travels with the delivery; a paper copy is signed and returned as proof of receipt. Hence the UBL Receipt Advice is not used.
The Seller raises the Invoice automatically when the despatch occurs, and the resolution of any shortages would be handled post-invoicing. The Invoice has to show the tax point date, the VAT (Value Added Tax) category to which the item belongs and also to show the VAT rate and total for each tax category on the invoice. VAT is also applied to charges such as the delivery surcharge. In order to encourage speedy payment of the amount due, the Seller offers a discount for prompt settlement, which the buyer can deduct if paying within 30 days. (Note that VAT regulations assume it will be taken and so the tax is calculated on the trade discounted total of line items plus any charges and less the settlement discount amount.)

Joinery Invoice - XML Instance, Joinery Invoice - printed version

This scenario is based on the products, product identification, business requirements and practices of a real UK joinery manufacturer and sales company. It operated its own specialised transport fleet delivering all over the United Kingdom and to offshore islands.

Appendix C. Formatting specifications for UBL document types

This collection contains examples of formatting specifications that can be followed to display instances of Universal Business Language (UBL) document types in human-readable form. Presentational semantics have not been formalized in this version of the UBL schema library, and they may never be formalized due to differing international requirements and conventions for the presentation of information found in business documents.

These specifications must not be considered as reference implementations of UBL or as normative components of the UBL specification; they are merely examples from one of what will probably be many available UBL stylesheet libraries.

The formatting specifications referenced below point to various layouts for the presentation of the information found in UBL instances. Some layouts are simplified presentations. Some layouts are intended to conform to the UN Layout Key for printed business documents, mimicking the intent of the UN Layout Key where official layouts do not currently exist.

The following collection of formatting specifications describes candidate renderings for the following UBL document types:

- UBL Order
- UBL Order Response
- UBL Order Response Simple
- UBL Order Change
- UBL Order Cancellation
- UBL Despatch Advice
- UBL Receipt Advice
- UBL Invoice

C.1 Documentation conventions

The following is an example of the documentation found in a formatting specification for a given field of a form on the rendered output.

C.1.2 Example form field information item documentation

Table1. XPath information

XPath addresses
`/po:Order/cat:BuyerParty/cat:Address/cat:Street`
`/po:Order/cat:BuyerParty/cat:Address/cat:Country/@countryId`

The box above includes two fictitious XML Path Language (XPath) addresses that documents the locations of information found in an XML instance. XPath addresses are used in XSLT stylesheets but can be used as above just for documentation because they are independent of the technology being used for transformation. The path is the route from the document element (the first step in the path) through to the information item actually being displayed.

In the first of the two examples above, the item being addressed is the `cat:Street` element that is a child of the `cat:Address` element. In the second of the two examples, the item being addressed is the `countryId` attribute of the `cat:Country` element.

The documented sections of the formatting specifications are oriented in the order of the fields found in the rendered result, approximately in the order of left to right from top to bottom (with some differences to accommodate logical groupings).

The formatting specifications are meant to be transformation technology agnostic. The specifications indicate what information goes where in the result, not how it gets there. Different implementations of transformation technologies can meet the need for the information found at the specified XPath address to appear at the specified location on the page.

C.2 Example implementations

These example implementations must not be considered as reference implementations of UBL formatting specifications or as normative components of the UBL delivery.

See `FS-implementations.html` for a list of known implementations of UBL Formatting Specifications at the time of publication.

C.3 Feedback

If you have any input to these formatting specifications, please do not hesitate to contact the UBL Forms Presentation Subcommittee following the directions on the home page cited above.

Appendix D. Tools and Deliverables

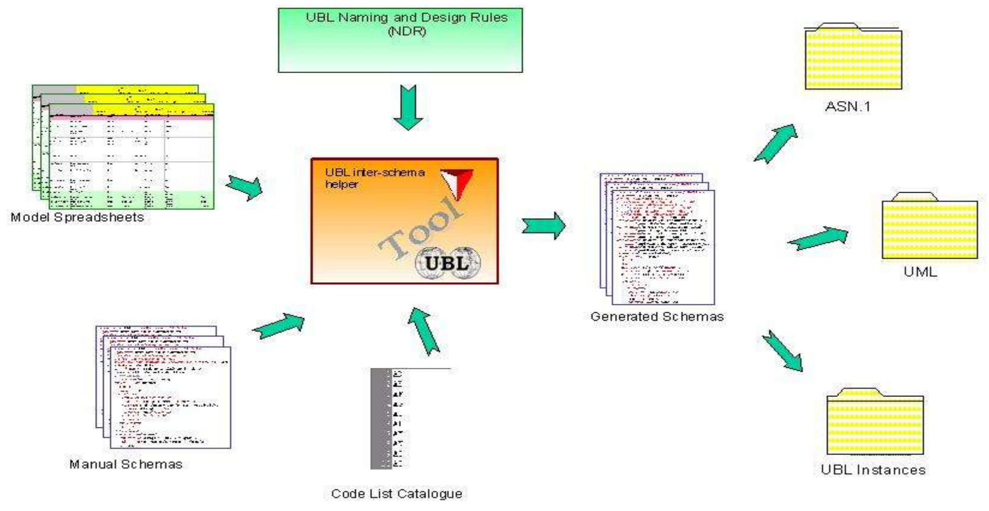


Figure 7. Tools and Deliverables

A variety of tools have been used in the generation of the UBL 1.0 Beta deliverables. Below we describe the main tools used to generate the normative schemas as well as the UML model diagrams and ASN.1 schemas.

D.1 Generation of Normative XSD Schemas

The Library Content Subcommittee (LCSC) has recognized the necessity of having a tool to automate the assembly of the various diversified input sources required for the generation of the UBL 1.0 schema sets. These diversified input sources are:

- LCSC data models represented in spreadsheets
- English prose descriptions of schema naming and design rules as developed by the UBL Naming and Design Rules Subcommittee
- 4 manually created XML schemas which are described at the beginning of the 'UBL Schemas', Section 4, of this document
- code list metadata captured in a Code List Catalogue spreadsheet

The diagram below illustrates the schema generation process that UBL has used:

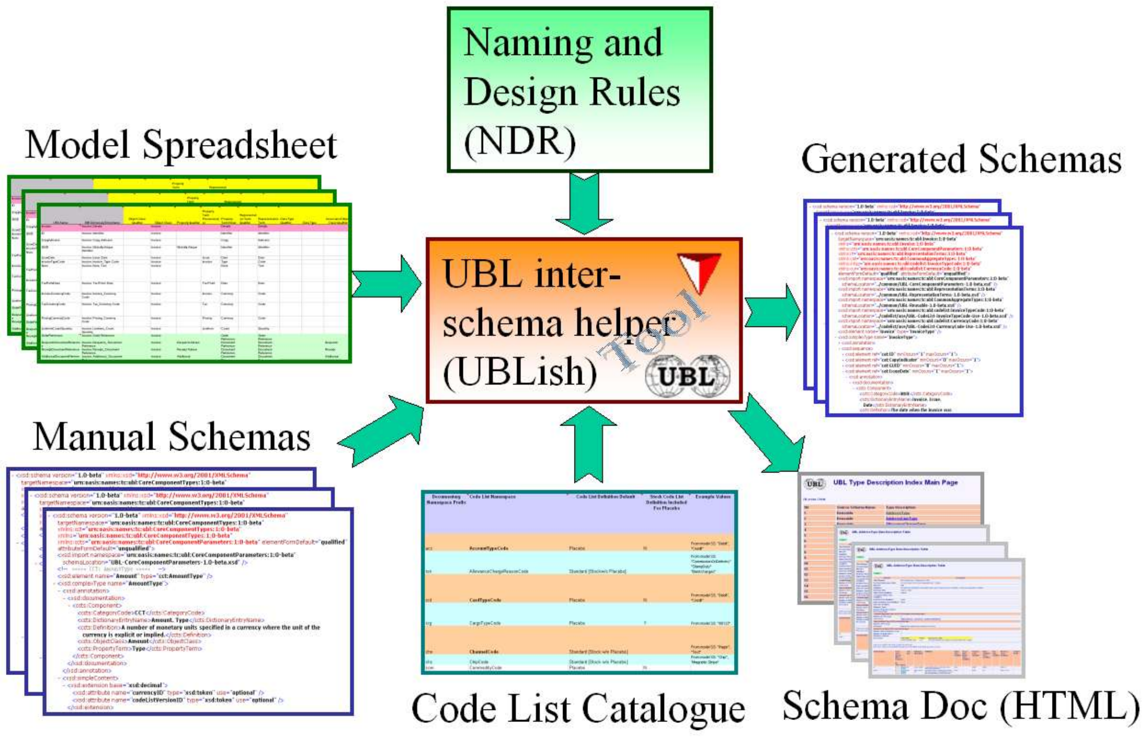


Figure 8. UBL Schema Generation Process

Central to generation of the UBL Library Schemas is the UBL inter-schema helper (UBLish) which combines and transforms all the input data sources and assembles them into the Generated Schemas shown on the right-hand-side of the diagram above. During the generation process, appropriate testing and validation of input data is done to ensure that data used for schema generation is proper and not propagated downstream. In addition, consistency checks, such as consistency amongst column relationships, consistency against NDR descriptions, etc are also done to increase the level of reliability and confidence in the generated schemas.

D1.1 UBL Schema Generation Process Inputs

D1.1.1 Model Spreadsheets

The design of the UBL Library model spreadsheets is intended primarily to capture the semantics of business interactions (see earlier sections in this document describing the Conceptual Model and Spreadsheets), but also supports the schema generation process by providing a specific, consistent format and positioning of this information which the schema generation tool can recognize. The tool depends on the format, location, and content of specific columns and cells to generate schemas that accurately represent the model described by the spreadsheets. There are 9 primary spreadsheets being utilized in this process: the Reusable spreadsheet, containing a collection of Aggregate Basic Information Entities (ABIEs) that are used throughout the other 8 models, and the 8 document model spreadsheets: Invoice, Order, OrderChange, OrderCancellation, OrderResponse, OrderResponseSimple, DespatchAdvice, and ReceiptAdvice.

D1.1.2 Manual Schemas

The Manual Schemas shown on the lower left of the diagram serve as input to the generation of the UBL Library document schemas described above, and represent the only schemas that are manually crafted and edited in UBL. There are 4 schemas that belong to this category:

CoreComponentParameters, CoreComponentTypes, RepresentationTerms and DataTypes. CoreComponentParameters defines the structure of metadata information that is used by all schemas delivered by UBL. The other 3 manually crafted schemas implement the Core Component Technical Specifications v2.0.

D1.1.3 Code List Catalogue Spreadsheet

The Code List Catalogue spreadsheet contains specific information used by the UBLish tool to produce UBL code list schemas. Namespace information in the Code List Catalogue is used to link the code list information to the data model, enabling the tool to generate main document schemas that utilize the code list schemas. With the help of UBLish, the laborious process of ensuring the definition of proper namespace values and schema locations of individual code list schemas vanishes because the generated schemas automatically will conform to XML Schema validation requirements.

D1.1.4 Naming and Design Rules

The UBL 1.0 Beta Naming and Design Rules (NDR) are serialized as an English prose document describing schema design guidelines such as to how XML tag names should be named, how schema type definitions should be structured, how the files could be named, how the namespace values would be composed, etc. Because of the prose nature of the NDR, this is a less straightforward component to implement. In practice, some of the guidelines go into constraining the values in the data model spreadsheets, while some of them go into the schema generation phase. All these positive definitive clauses and constraint-oriented guidelines are transformed and implemented in various parts of the UBLish logic that governs the form and shape of the generated schemas.

D.1.2 UBLish

The schema generator – UBL inter-schema helper (UBLish) – is not included in the deliverable package. This is because the application is developed and owned by SoftML and could not be packaged into the main UBL release as part of OASIS property. However, SoftML has since March 2003 made available its UBLish (for 0p70 release of UBL), and will be again making the upgraded version designed for UBL 1.0 release on its website. The UBLish application is royalty free and is available for download at SoftML website at:

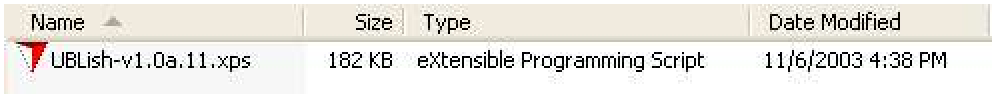
http://SoftML.Net/jedi/ubl/sw/UBLish

Installation instructions and usage notes are found on the URL indicated. Basically, the UBLish is programmed in XPS (eXtensible Programming Script). To execute UBLish, one would need to first install the public version of the XPS run-time integration engine, which is also available from SoftML website at:

http://SoftML.Net/xps/

Installation should be quite straightforward. Both components need to be installed before UBLish can perform its functions. The public version of XPS run-time integration engine is also royalty free, but has separate licensing terms that is more commercial in nature. Users of public version of XPS run-time integration engine should not expect any support other than information that is released on the website.

Once the run-time integration engine and the UBLish are installed, you should see something like the following snapshot in your directory viewer:



Name	Size	Type	Date Modified
UBLish-v1.0a.11.xps	182 KB	eXtensible Programming Script	11/6/2003 4:38 PM

At this point, double click on the inverted 3D prism icon to run UBLish.

D.1.2.2 Use of UBLish

One might ask why one would have the need to check out UBLish, or even try running it, since it has already produced the normative UBL Schemas, and by itself is a non-normative item.

However, serious users would quickly find the need to look at the magic box in the middle of the diagram “UBL Schema Generation Process” to understand what went on in the whole UBL machinery that has output the schemas. Being written in XPS scripting language, UBLish allows the user to examine the functions and variable assignments easily since the script itself is the executable. It therefore provides another aspect of documentation in and by itself regarding how UBL manages various sources of input requirements in the process of generating the schemas.

Another group of users might also be expected to download and install UBLish – users who are looking at customizing UBL and borrowing the same machinery that generated UBL schemas in their local environments. This group of users may or may not want to understand how UBLish works. But by installing UBLish and modifying the spreadsheets with their own modeling data, they gain a machinery that can immediately output UBL-look-alike schemas in a quick and efficient manner.

D.1.2.3 UBLish+ Extension

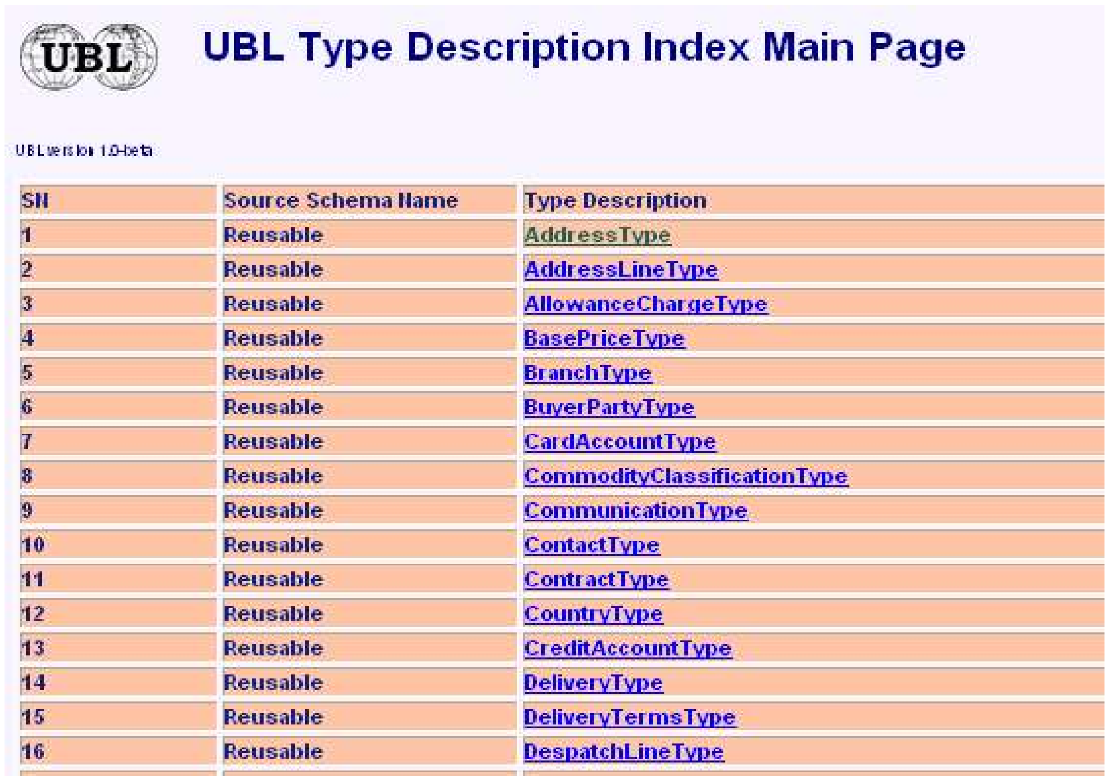
SoftML internally continues its ad hoc and experimental extensions to UBLish. Some special functions had generated derivative information that has helped in providing corrective information to UBL schema and modeling design process, while other functions had resulted in enhanced views, functionalities and other aspects of schema uses. Yet other functions are temporal in nature, and get changed as design rules change or when inter-schema architectural decisions get altered. All these varying features and functionalities are grouped under a UBLish+ Extension module that SoftML does not release.

D.1.2.4 Schema Documentation

One of the by-products of UBLish+ Extension is the Schema Documentation HTML set of files. The set of files is also made available at SoftML website at:

http://SoftML.Net/jedi/ubl/

The main index page is as shown below:



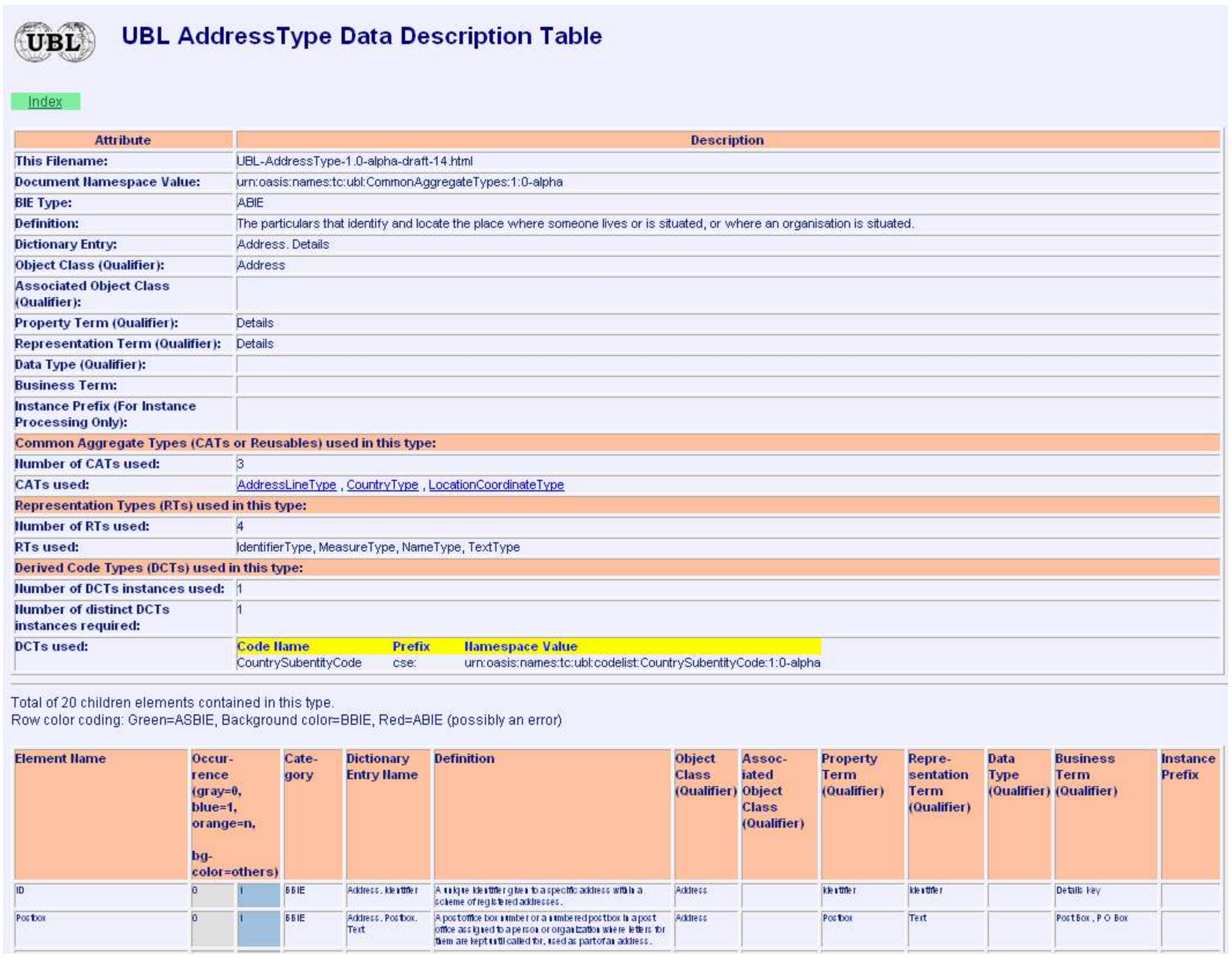
UBL Type Description Index Main Page

UBL version 1.0-beta

SN	Source Schema Name	Type Description
1	Reusable	AddressType
2	Reusable	AddressLineType
3	Reusable	AllowanceChargeType
4	Reusable	BasePriceType
5	Reusable	BranchType
6	Reusable	BuyerPartyType
7	Reusable	CardAccountType
8	Reusable	CommodityClassificationType
9	Reusable	CommunicationType
10	Reusable	ContactType
11	Reusable	ContractType
12	Reusable	CountryType
13	Reusable	CreditAccountType
14	Reusable	DeliveryType
15	Reusable	DeliveryTermsType
16	Reusable	DespatchLineType

Basically, the user starts with browsing this “index.html” page and gets presented with a listing of all the ABIE types defined in UBL schemas, including all ABIE types defined in the Reusable and all 8 document schemas.   On clicking any of these types, the user is hyperlinked into the particular page containing intimate details related to that type.

For instance, if the user clicks on the “AddressType” hyperlink, the screen will show the following color-coded page of information regarding the ABIE type “AddressType”:


UBL AddressType Data Description Table

Index

Attribute	Description
This Filename:	UBL-AddressType-1.0-alpha-draft-14.html
Document Namespace Value:	urn:oasis:names:tc:ubl:CommonAggregateTypes:1:0-alpha
BIE Type:	ABIE
Definition:	The particulars that identify and locate the place where someone lives or is situated, or where an organisation is situated.
Dictionary Entry:	Address. Details
Object Class (Qualifier):	Address
Associated Object Class (Qualifier):	
Property Term (Qualifier):	Details
Representation Term (Qualifier):	Details
Data Type (Qualifier):	
Business Term:	
Instance Prefix (For Instance Processing Only):	
Common Aggregate Types (CATs or Reusables) used in this type:	
Number of CATs used:	3
CATs used:	AddressLineType , CountryType , LocationCoordinateType
Representation Types (RTs) used in this type:	
Number of RTs used:	4
RTs used:	IdentifierType, MeasureType, NameType, TextType
Derived Code Types (DCTs) used in this type:	
Number of DCTs instances used:	1
Number of distinct DCTs instances required:	1
DCTs used:	Code Name: CountrySubentityCode; Prefix: cse:; Namespace Value: urn:oasis:names:tc:ubl:codelist:CountrySubentityCode:1:0-alpha

Total of 20 children elements contained in this type.
Row color coding: Green=ASBIE, Background color=BBIE, Red=ABIE (possibly an error)

Element Name	Occurrence (gray=0, blue=1, orange=n, bg-color=others)		Category	Dictionary Entry Name	Definition	Object Class (Qualifier)	Associated Object Class (Qualifier)	Property Term (Qualifier)	Representation Term (Qualifier)	Data Type (Qualifier)	Business Term (Qualifier)	Instance Prefix
ID	0	1	BBIE	Address. Identifier	A unique identifier given to a specific address within a scheme of registered addresses.	Address		Identifier	Identifier		Details Key	
Postbox	0	1	BBIE	Address. Postbox. Text	A post office box number or a numbered postbox in a post office assigned to a person or organization where letters for them are kept until called for, used as part of an address.	Address		Postbox	Text		Post Box , P O Box	

Not only does it show the individual metadata components from which the original modeling spreadsheet was taken to generate the type, there are also listings of which other Reusable types as well as which other code list (schema) types are being used by the selected type.

Through the web of hyperlinks, user can then navigate and explore from here further sub-types directly without going back to the main page again.

D.2 Generation of Non-Normative Components

D2.1 Generation of UML Models

Ontogenics Corporation's *hyper*Model tool was used during development of the UBL library specification to automatically transform the normative XML Schemas into a UML implementation model. The class diagrams in the UBL 1.0 Beta release were generated from that implementation model. *hyper*Model enables round-trip transformation between any XML Schema and any UML class model. The UML profile used to guide mapping to/from XML Schema enables complete access to the features of the XSD language. For example, you can customize or extend the UBL library implementation model in UML, then generate a new set of schemas for your extensions that reuse the UBL library components. Class diagrams are created using an approach similar to web browsers; you can explore the structure of complex models, either imported from XML

Schemas or created directly in UML. *hyper*Model is designed as a plug-in to the Eclipse IDE, so these features can be used alone or integrated with other plug-ins used within the same desktop IDE.

D2.2 Generation of Abstract Syntax Notation One (ASN.1) Conformant Schemas

The ASN.1 schemas for UBL were created by using a tool from OSS Nokalva (www.oss.com) that conforms to ITU-T Recommendation X.694 | ISO/IEC 8825-5 for converting XML Schema to ASN.1. After feeding the UBL XSD to the OSS Nokalva XSD to ASN.1 conversion tool, the generated ASN.1 was fed to the PrettyPrint tool at http://asn1.elibel.tm.fr website to produce the nicely formatted HTML version of the UBL ASN.1 schemas.

Appendix E. ASN.1 Materials [informative]

ASN.1 Specification of UBL

UBL also provides an ASN.1 specification for UBL messages that provides an alternative XML schema definition for the XML documents. This ASN.1 specification defines the same valid XML documents as the XSD Schema, which is the primary definition of valid XML documents. Use of this ASN.1 XML schema enables ASN.1 tools to be used for UBL transfers, and in conjunction with the ASN.1 Packed Encoding Rules, provides a specification for an efficient "binary XML" encoding of UBL messages.

This is the definition of binary XML encodings of UBL messages.

The ASN.1 definition for the current release of UBL can be found at:

asn/asn1-UBL-beta-1.0.html

ASN.1 References

[ASN.1] Abstract Syntax Notation One, ITU-T Recommendation | ISO/IEC International Standard

http://www.itu.int/ITU-T/studygroups/com17/languages

1156 # Appendix F. Code List Schemas

1157

codelist/placebo/	codelist/use/
UBL-CodeList-AccountTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-AccountTypeCode-Use-1.0-beta.xsd
UBL-CodeList-AllowanceChargeReasonCode-Placebo-1.0-beta.xsd	UBL-CodeList-AllowanceChargeReasonCode-Use-1.0-beta.xsd
UBL-CodeList-CardTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-CardTypeCode-Use-1.0-beta.xsd
UBL-CodeList-CargoTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-CargoTypeCode-Use-1.0-beta.xsd
UBL-CodeList-ChannelCode-Placebo-1.0-beta.xsd	UBL-CodeList-ChannelCode-Use-1.0-beta.xsd
UBL-CodeList-ChipCode-Placebo-1.0-beta.xsd	UBL-CodeList-ChipCode-Use-1.0-beta.xsd
UBL-CodeList-CommodityCode-Placebo-1.0-beta.xsd	UBL-CodeList-CommodityCode-Use-1.0-beta.xsd
UBL-CodeList-ContractTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-ContractTypeCode-Use-1.0-beta.xsd
UBL-CodeList-CoordinateSystemCode-Placebo-1.0-beta.xsd	UBL-CodeList-CoordinateSystemCode-Use-1.0-beta.xsd
UBL-CodeList-CountryIdentificationCode-Placebo-1.0-beta.xsd	UBL-CodeList-CountryIdentificationCode-Use-1.0-beta.xsd
UBL-CodeList-CountrySubentityCode-Placebo-1.0-beta.xsd	UBL-CodeList-CountrySubentityCode-Use-1.0-beta.xsd
UBL-CodeList-CurrencyCode-Placebo-1.0-beta.xsd	UBL-CodeList-CurrencyCode-Use-1.0-beta.xsd
UBL-CodeList-DespatchAdviceTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-DespatchAdviceTypeCode-Use-1.0-beta.xsd
UBL-CodeList-DispositionCode-Placebo-1.0-beta.xsd	UBL-CodeList-DispositionCode-Use-1.0-beta.xsd
UBL-CodeList-DocumentStatusCode-Placebo-1.0-beta.xsd	UBL-CodeList-DocumentStatusCode-Use-1.0-beta.xsd
UBL-CodeList-EmergencyCardCode-Placebo-1.0-beta.xsd	UBL-CodeList-EmergencyCardCode-Use-1.0-beta.xsd
UBL-CodeList-EmergencyProceduresCode-Placebo-1.0-beta.xsd	UBL-CodeList-EmergencyProceduresCode-Use-1.0-beta.xsd
UBL-CodeList-ExemptionReasonCode-Placebo-1.0-beta.xsd	UBL-CodeList-ExemptionReasonCode-Use-1.0-beta.xsd
UBL-CodeList-FromEventCode-Placebo-1.0-beta.xsd	UBL-CodeList-FromEventCode-Use-1.0-beta.xsd
UBL-CodeList-FullnessIndicationCode-Placebo-1.0-beta.xsd	UBL-CodeList-FullnessIndicationCode-Use-1.0-beta.xsd
UBL-CodeList-HandlingCode-Placebo-1.0-beta.xsd	UBL-CodeList-HandlingCode-Use-1.0-beta.xsd
UBL-CodeList-HazardousPackingCriteriaCode-Placebo-1.0-beta.xsd	UBL-CodeList-HazardousPackingCriteriaCode-Use-1.0-beta.xsd
UBL-CodeList-InhalationToxicityZoneCode-Placebo-1.0-beta.xsd	UBL-CodeList-InhalationToxicityZoneCode-Use-1.0-beta.xsd
UBL-CodeList-InvoiceTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-InvoiceTypeCode-Use-1.0-beta.xsd
UBL-CodeList-IssuerTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-IssuerTypeCode-Use-1.0-beta.xsd
UBL-CodeList-LatitudeDirectionCode-Placebo-1.0-beta.xsd	UBL-CodeList-LatitudeDirectionCode-Use-1.0-beta.xsd
UBL-CodeList-LineStatusCode-Placebo-1.0-beta.xsd	UBL-CodeList-LineStatusCode-Use-1.0-beta.xsd
UBL-CodeList-LocaleCode-Placebo-1.0-beta.xsd	UBL-CodeList-LocaleCode-Use-1.0-beta.xsd
UBL-CodeList-LongitudeDirectionCode-Placebo-1.0-beta.xsd	UBL-CodeList-LongitudeDirectionCode-Use-1.0-beta.xsd
UBL-CodeList-MedicalFirstAidGuideCode-Placebo-1.0-beta.xsd	UBL-CodeList-MedicalFirstAidGuideCode-Use-1.0-beta.xsd
UBL-CodeList-NatureCode-Placebo-1.0-beta.xsd	UBL-CodeList-NatureCode-Use-1.0-beta.xsd
UBL-CodeList-OrderAcknowledgementCode-Placebo-1.0-beta.xsd	UBL-CodeList-OrderAcknowledgementCode-Use-1.0-beta.xsd
UBL-CodeList-PaymentChannelCode-Placebo-1.0-beta.xsd	UBL-CodeList-PaymentChannelCode-Use-1.0-beta.xsd
UBL-CodeList-PaymentMeansTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-PaymentMeansTypeCode-Use-1.0-beta.xsd
UBL-CodeList-PeriodDescriptionCode-Placebo-1.0-beta.xsd	UBL-CodeList-PeriodDescriptionCode-Use-1.0-beta.xsd

codelist/placebo/	codelist/use/
UBL-CodeList-PositionCode-Placebo-1.0-beta.xsd	UBL-CodeList-PositionCode-Use-1.0-beta.xsd
UBL-CodeList-PriorityLevelCode-Placebo-1.0-beta.xsd	UBL-CodeList-PriorityLevelCode-Use-1.0-beta.xsd
UBL-CodeList-RateCategoryCode-Placebo-1.0-beta.xsd	UBL-CodeList-RateCategoryCode-Use-1.0-beta.xsd
UBL-CodeList-RegulationCode-Placebo-1.0-beta.xsd	UBL-CodeList-RegulationCode-Use-1.0-beta.xsd
UBL-CodeList-RejectActionCode-Placebo-1.0-beta.xsd	UBL-CodeList-RejectActionCode-Use-1.0-beta.xsd
UBL-CodeList-RejectReasonCode-Placebo-1.0-beta.xsd	UBL-CodeList-RejectReasonCode-Use-1.0-beta.xsd
UBL-CodeList-RiskResponsibilityCode-Placebo-1.0-beta.xsd	UBL-CodeList-RiskResponsibilityCode-Use-1.0-beta.xsd
UBL-CodeList-SalesConditionsActionCode-Placebo-1.0-beta.xsd	UBL-CodeList-SalesConditionsActionCode-Use-1.0-beta.xsd
UBL-CodeList-SealStatusCode-Placebo-1.0-beta.xsd	UBL-CodeList-SealStatusCode-Use-1.0-beta.xsd
UBL-CodeList-ShortageActionCode-Placebo-1.0-beta.xsd	UBL-CodeList-ShortageActionCode-Use-1.0-beta.xsd
UBL-CodeList-SubstitutionStatusCode-Placebo-1.0-beta.xsd	UBL-CodeList-SubstitutionStatusCode-Use-1.0-beta.xsd
UBL-CodeList-TaxLevelCode-Placebo-1.0-beta.xsd	UBL-CodeList-TaxLevelCode-Use-1.0-beta.xsd
UBL-CodeList-TaxTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-TaxTypeCode-Use-1.0-beta.xsd
UBL-CodeList-TimingComplaintCode-Placebo-1.0-beta.xsd	UBL-CodeList-TimingComplaintCode-Use-1.0-beta.xsd
UBL-CodeList-TransitDirectionCode-Placebo-1.0-beta.xsd	UBL-CodeList-TransitDirectionCode-Use-1.0-beta.xsd
UBL-CodeList-TransportEquipmentSizeTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-TransportEquipmentSizeTypeCode-Use-1.0-beta.xsd
UBL-CodeList-TransportEquipmentTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-TransportEquipmentTypeCode-Use-1.0-beta.xsd
UBL-CodeList-TransportMeansTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-TransportMeansTypeCode-Use-1.0-beta.xsd
UBL-CodeList-TransportModeCode-Placebo-1.0-beta.xsd	UBL-CodeList-TransportModeCode-Use-1.0-beta.xsd
UBL-CodeList-UNDGCode-Placebo-1.0-beta.xsd	UBL-CodeList-UNDGCode-Use-1.0-beta.xsd
UBL-CodeList-UnitTypeCode-Placebo-1.0-beta.xsd	UBL-CodeList-UnitTypeCode-Use-1.0-beta.xsd