Universal Business Language (UBL) Code List Representation

Version: 1.1 draft 17 January 2005

Document identifier:

wd-ublclsc-codelist-20050103.doc

Location:

http://www.oasis-open.org/committees/ubl/

Editor:

Marty Burns for National Institute of Standards and Technology, NIST, burnsmarty@aol.com

Contributors:

Anthony Coates abcoates@londonmarketsystems.com
Mavis Cournane mavis.cournane@cognitran.com
Suresh Damodaran Suresh_Damodaran@stercomm.com
Anne Hendry anne.hendry@sun.com
G. Ken Holman gkholman@CraneSoftwrights.com
Serm Kulvatunyou serm@nist.gov
Eve Maler eve.maler@sun.com
Tim McGrath tmcgrath@portcomm.com.au
Mark Palmer mark.palmer@nist.gov
Sue Probert sue.probert@dial.pipex.com
Lisa Seaburg lseaburg@aeon-llc.com
Paul Spencer paul.spencer@boynings.co.uk
Alan Stitzer alan.stitzer@marsh.com
Frank Yang Frank.Yang@RosettaNet.org

Abstract:

This specification provides rules for developing and using reusable code lists. This specification has been developed for the UBL Library and derivations thereof, but it may also be used by other technologies and XML vocabularies as a mechanism for sharing code lists and for expressing code lists in W3C XML Schema form.

Note: This draft is an intermediate edit along the path of UBL 1.1. The present revision has only modified the front matter of the document and revised the requirements. The reader is directed to ignore the balance of the draft contained herein.

Status:

This document was developed by the OASIS UBL Code List Subcommittee **[CLSC]**. Your comments are invited. Members of this subcommittee should send comments on this

specification to the ubl-clsc@lists.oasis-open.org list. Others should subscribe to and send comments to the ubl-comment@lists.oasis-open.org list.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights (OASIS-IPR) section of the Security Services TC web page (http://www.oasis-open.org/who/intellectualproperty.php

Table of Contents

1 Introduction

Trading partners utilizing the Universal Business Language (UBL) must agree on restricted sets of coded values, termed "code lists", from which values populate particular UBL data fields. Code lists are accessed using many technologies, including databases, programs and XML. Code lists are expressed in XML for UBL using W3C XML Schema for authoring guidance and processing validation purposes.

It is important to note that XML schema languages are not purely abstract data models. They provide only a particular representation of the data. In addition, there are many roughly equivalent design choices (e.g. elements versus attributes). The underlying logical model is obscured, and can be difficult to extract. Therefore, XML schema languages are principally useful as a way of specifying rules to an XML validation engine. Database schemas and programming language class models would have their own specific representations of the logical data models.

A good logical data model format should allow the information about code lists to be expressed in a format that is as simple and unambiguous as possible. To maximize the abstraction on one hand, and the utility of the code list representations on the other, this document first derives an abstract data model of a code list, and then, an XMLSchema representation of that data model.

Note that there are two major aspects of a model of code lists – the list of codes and descriptive information about the code list termed “supplementary components”. Supplementary components include information such as origin and version, for example. Supplementary components describe the metadata about the code lists and codes themselves. They appropriately describe the context within which individual codes can be understood.

The document begins with a section expositing the requirements adopted by the committee in order to make certain that design follows requirements. These requirements were used to steer the design choices elected in the balance of the document.

This specification was developed by the OASIS UBL Code List Subcommittee **[CLSC]** to provide rules for developing and using reusable code lists expressed using W3C XML Schema **[XSD]** syntax.

The contents combine requirements and solutions previously developed by UBL’s Library, Naming, and Design Rules subcommittee **[CL5]**, the work of the National Institute of Standards “eBusiness Standards Convergence Forum” **[eBSC]** with contributions from Frank Yang and Suresh Damodaran of Rosettanet **[eBSCMemo]**, and position papers by Anthony Coates **[COATES]**, Gunther Stuhec **[STUHEC]**, and Paul Spencer **[SPENCER]**.

The data model attempts to be sufficiently general to be employable with other technologies (e.g. non-XML) and in other scenarios that are outside the scope of this committee's work.

This specification is organized as follows:

- Section 2 provides requirements for code lists;
- Section 3 provides a data and metadata model (supplementary components) of code lists;
- Section 4 is an XMLSchema representation of the model;
- Section 5 is the recommendations for code producers and the compliance rules.

1.1 About the current version

The Code List model described in this paper for UBL 1.0 has laid much of the groundwork for extensible code lists. It includes an extensibility mechanism based on XSD substitution groups that has not been adopted for UBL 1.0 but will serve as a starting point for work on a code list extension mechanism for

UBL 1.1. The current specification places a priority on uniformity of code list metadata independent of the mechanism eventually adopted for code list extension.

The UBL team has embarked on an effort, in conjunction with NIST's eBusiness Standards Convergence Forum (eBSC) to fulfill the goals of constructing a code list model that can be reused throughout industry. The current version contains an update to the descriptions of the requirements and some enhanced requirements discovered in the interim. For the time being, those features beyond the UBL1.0 are still labeled as FUTURE such designation to be removed further along in the version 1.1 process.

Persons wishing to engage in the further evolution of this specification are urged to join the OASIS Universal Business Language Technical Committee (http://oasis-open.org/).

1.2 Scope and Audience

The rules in this specification are designed to encourage the creation and maintenance of code list modules by their proper owners as much as possible. It was originally developed for the UBL Library and derivations thereof, but it is largely not specific to UBL needs; it may also be used with other XML and non-XML vocabularies as a mechanism for sharing code lists. If enough code-list-maintaining agencies adhere to these rules, we anticipate that a more open marketplace in XML-encoded code lists will emerge for all XML vocabularies. In addition, it is anticipated that these common definitions will find use in other non-XML applications that need to store or otherwise represent the same data as it traverses from application to application.

This specification assumes that the reader is familiar with the UBL Library and with the ebXML Core Components **[CCTS2.01]** concepts and ISO 11179 **[ISO 11179]** concepts that underlie it. While mastery of these concepts is not essential to the understanding and use of this document, they are useful in explaining the concepts behind the organization and structure of this material.

1.3 Terminology and Notation

The text in this specification is normative for UBL Library use unless otherwise indicated. The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this specification are to be interpreted as described in **[RFC2119]**.

Terms defined in the text are in **bold**. Refer to the UBL Naming and Design Rules **[NDR]** for additional definitions of terms.

Core Component names from ebXML are in *italic*.

```
Example code listings appear like this.
```

Note: Non-normative notes and explanations appear like this.

Conventional XML namespace prefixes are used throughout this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

The prefix `xs:` stands for the W3C XML Schema namespace **[XSD]**.

The prefix `xhtml:` stands for the XHTML namespace.

The prefix `iso3166:` stands for a namespace assigned by a fictitious code list module for the ISO 3166-1 country code list.

1.3.1 Definitions

[1/2/05 MJB] Need to substantially populate this list with all acronyms and terminology used in this paper.

BIE	Business Information Entities.
code	A group of contiguous text characters that together uniquely specify the name and/or attributes of a particular field or “element” embedded in a stream of data.
code list	A set containing one or more codes or code values that is associated with one or more elements of data stream.
code list mechanism	A term used to distinguish this specification from the instances of actual code lists based on it.
code value	See the definition for “code” above.
core components	A building block such as account identification data that contains pieces of business information associated with a single concept. Core components are sufficiently general to be used across several or many different business sectors.
data model	A technique, set of rules and/or methods used to organize information objects and thereby define a structure for data. Such models are created to streamline the storage/retrieval, manipulation, use or comprehension of data and/or to provide information about its interrelationships, meaning, function or usage.
ebXML	An acronym for Electronic Business using eXtensible Markup Language. ebXML is a modular suite of specifications that enables enterprises at disparate geographical locations to conduct business over the Internet. ebXML, provides standard methods for exchanging business messages, for conducting trading relationships, for communicating data in common terms and for defining and registering business processes.
enumeration	A list or set, usually containing two or more entries, of associated data elements. Entries have been logically grouped or associated, and possibly named as a set, permitting later selection of a single member or entry for a purpose such as specifying the characteristics of an object.
ISO 11179	An International Organization for Standardization specification that provides rules and guidelines for the naming, definition, creation and registration of data elements. It also contains information about the type of metadata that should be specified for data elements.
metadata	Information, for example characteristics, content, context or structure, that is associated with a data object. In short, metadata is “data about data.”
NDR	Naming and design rules.
OASIS	An acronym for Organization for the Advancement of Structured Information Standards. OASIS is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. It produces standards used for or by Web services, security, e-business, the public sector and application-specific markets.
Perl	The Practical Extraction and Report Language is an interpreted programming language that utilizes features from C, sed, awk, and sh. It scans arbitrary text files, extracts information from them and generates output based on the extracted information. As open source software its source code is available.

supplementary components	Supplementary components describe the metadata about the code lists and codes themselves. They appropriately describe the context within which individual codes can be understood.
UBL	A generic XML interchange format for business documents that can be extended to meet the requirements of particular industries. The UBL specification currently consists of a library of XML schemas for reusable data components (e.g. “address”, “payment”, etc.), a set of XML schemas for common business documents (e.g. “Order”, “Invoice”, etc.) and support for industry-specific extensions to the format.
URI	An acronym for Uniform Resource Identifier. Each URI is a unique identifier for a resource or object on the Internet. URIs are drawn from a universal set of names or addresses and the objects or resources to which they refer can be accessed with well-known protocols. Every URI is located in one or more registries of such names and/or addresses. A Uniform Resource Locator (URL) is one example of a URI.
W3C	An acronym for World Wide Web Consortium. This organization develops and distributes information, specifications, guidelines, software, and tools that enhance the operation of the Internet. It also acts as a forum for commerce, education and communication.
XML	An acronym for eXtensible Markup Language. XML is a set of rules for the creation of customized markup languages that are used in textual documents to name, describe the attributes of and specify the relationships between data elements contained in those documents. XML is derived from SGML (Standard Generalized Markup Language) and has been designed for the transport and sharing of data..
XML Schema	A textual description of the appearance, interrelationships and valid value ranges for the data elements in an XML stream.

2 Requirements for Code Lists

"There can be no solution without a requirement!"

This section summarizes the requirements to be addressed by this paper. Requirements are identified in the heading for each one as: [Rn], where 'n' is the requirement number. This draft contains requirements that have been accumulated for code lists in general. In order to allow for the interim publishing of this specification, several of the requirements have been labeled as future requirements: [Rn (Future)]

[3/9/04 MJB] The requirements in this section need to be associated ultimately with the design in sections 3 and 4. This will be done by listing requirements addressed in each subsection below the subsection title line.

2.1 Overview

The goal of this document is to provide a representation model or mechanism for code lists that are extensible, restrictable, traceable, and cognizant of the need for code lists to be maintained by various organizations who are authorities on their content. Code lists developed bye this means will be infused with the requirements outlined in this section.

Note that the code list *mechanism* of this specification needs to support all of the requirements in this section. However, any single code list based on this specification may not be required to meet all requirements simultaneously. The appropriate subset of requirements that a given code list must support is summarized in the use cases presented in the conformance section (5 Conformance to UBL Code Lists).

2.2 Use and management of Code Lists

This section describes requirements for the use and management of code lists.

2.2.1 [R1] First-order business information entities

Code list values may appear as first-order business information entities (BIEs). For example, one property of an address might be a code indicating the country. This information appears in an element, according to the Naming and Design Rules specification **[NDR]**. For example, in XML a country code might appear as:

```
<Country>UK</Country>
```

2.2.2 [R2] Second-order business information entities

Code list values may appear as second-order information that qualifies another BIE. For example, any information of the Amount core component type must have a supplementary component (metadata) indicating the currency code. For example, in XML a currency code might appear as an attribute – the value of element Currency is 2456000; the code EUR describes that these are in Euros:

```
<Currency code="EUR">2456000</Currency>
```

2.2.3 [R3] Data and Metadata model separate from Schema representation

Since all uses of code lists will not be exclusively within the XML domain – ie. Databases, etc…, it is desirable to separate the description of the data model from its XML representative form. This will facilitate use for other purposes of the semantically identical information.

Code list interoperability comes about when different specifications or applications use the same enumerated values (or aliases thereof) to represent the same things/concepts/etc. Sharing XML schemas (or fragments) is one way of achieving this, but it is not a necessary method for achieving this goal.

Broader interoperability can be achieved instead by defining a format which models code lists independently of any validation or representation mechanisms that they may be used with. Such a data model should be able to be processed to produce the required XML Schemas, and should also be able to be processed to produce other artifacts, e.g. Java type-safe enumeration classes, database Schemas, code snippets for HTML forms or XForms, etc.

The format should be appropriate for use across a range of standards activities, i.e. it should embody the most generic view of code lists, and not any particular group's specific view. It should also be useful for implementations of those standards, not just for the standards activity itself.

2.2.4 [R4] XML and XML Schema representation

The principal anticipated use of the code list model will be in XML applications – XML for usage, and XMLSchema for validation of instance documents. This paper should realize a proper XML / XMLSchema representation for the code list model.

2.2.5 [R5 (Future)] Machine readable data model

A data model is an abstraction and it must be converted to explicit representation for use. The principal such use anticipated by this effort is that of XML data exchange. A machine readable representation of the data model makes the lossless transfer of all meaning to the representation of choice easier since it can be automated. It is therefore desirable that the data model be expressed in a machine-readable form. By lossless transfer it is intended that once a transfer of a code list model into an alternate form, all original information or semantics is contained in the alternate for so that the original could then be recreated solely from the contents of the original form.

By way of a negative example, consider the following translation that is not lossless:

> Assume that a number represented in syntax A 98.6. Syntax B is restricted by its designers to only integral number representations. Thus the translation of 98.6 would result in 98. Clearly, the translation was not lossless since the fractional part (although not needed by applications using Syntax B) was truncated. There is no way to deduce 98.6 solely from the number 98.

2.2.6 [R6 (Future)] Conformance test for code lists

An abstract model for code lists requires a method to ensure conformance and consistency of the rendering of instance Schemas based on the model. There shall be a definition of this conformance to qualify the results of the usage of this specification.

2.2.7 [R6a] Supplementary components or metadata available in instance documents

Instance documents often have fiduciary requirements. This requirement is independent of the need to be able to validate contents according to a referenced schema. This requires that some meta-information be explicitly contained in the instance document, irrespective of its availability in a referenced document. Therefore:

- The supplementary components of the code lists of code list values utilized in a UBL instance shall optionally be available in the XML instance proper without any processing from any external source including any schema expression.

- The supplementary components shall be optionally available for all code-list-value information items even when two or more such information items are found in the set of data and attribute information items for any given element.

2.3 Types of code lists

2.3.1 [R7] *UBL maintained Code List*

UBL will make use of code lists that describe information content specific to UBL. Such code lists are intended to become part of the UBL Library of schemas.

In some cases the UBL Library may have to be extended to meet specific business requirements. In other cases where a suitable code list does not exist in the public domain, that code list and all its values may have to be added to the UBL Library where it will be maintained. Both of these types of code lists would be considered UBL-internal code lists.

2.3.2 [R8] *Identify and use external standardized code lists*

Because the majority of code lists are expected to be owned and maintained by external agencies, UBL shall make maximum use of such external code lists where they exist. The UBL Library SHOULD identify and use external standardized code lists rather than develop its own UBL-native code lists.

2.3.3 [R9] *Private use code list*

This model must support the construction of private code lists where an existing external code list needs to be extended, or where no suitable external code list exists.

2.4 Technical requirements of Code Lists

Following are technical quality requirements for code lists.

2.4.1 [R10] Semantic clarity

The ability to “de-reference” the ultimate normative definition of the code being used. The supplementary components for “Code.Type” CCTs are the expected way of providing this clarity, but there are many ways to supply values for these components in XML, and it’s even possible to supply values in some non-XML form that can then be referenced by the XML form.

[1/1/05 MJB] Still need to elaborate this requirement.

2.4.2 [R11] Interoperability

Interoperability can be thought of as the sharing of a common understanding of the limited set of codes expected to be used. There is a continuum of possibilities here. For example, a schema datatype that allows only a hard-coded enumerated list of code values provides “hard” (but inflexible) interoperability. On the other hand, merely documenting the intended shared values is more flexible but somewhat less interoperable, since there are fewer penalties for private arrangements that go outside the standard boundaries. This requirement is related to, but distinct from, validatability and context rules friendliness

2.4.3 [R12] External maintenance

The ability for non-UBL organizations to create XSD schema modules that define code lists in a way that allows UBL to reuse them without modification on anyone's part. Some standards bodies are already doing this, although we recognize that others may never choose to create such modules.

2.4.4 [R13] Validatability

The ability to use XSD (or correspondingly suitable tool if not an XML based representation of the code list) to validate that a code appearing in an instance is legitimately a member of the referenced code list.

2.4.5 [R14] Context rules friendliness

The code list mechanism shall use expected normal mechanisms of the UBL Naming and Design Rules (NDR) without unnecessarily adding custom features just for code lists.

[1/3/05 MJB] Note: If any extension is necessary or agreed upon, changes in the NDR shall be required to evidence it.

2.4.6 [R15] Upgradability / Extensibility without modifying underlying references

The code list mechanism shall support the ability to begin using a new version of a code list without the need for upgrading, modifying, or customizing the source schema modules (or other original referenced material).

It is therefore necessary to establish a mechanism by which a given code or code list can be extended for use without having to alter the underlying source material. When such a extension is made, it is also necessary to be able to determine unambiguously the nature and source of the modification so that its use can be validated.

2.4.7 [R16] Readability

A representation in the XML instance that provides code information in a clear, easily readable form. For example, representing codes as a sequence of arbitrary number sequences would fail this test as there would be no contextual information.

2.4.8 [R17] Code lists must be unambiguously identified

The generation of multiple versions of a code list and the coexistence of more than one version shall be supported. The procedure used to generate each such revision from an earlier version shall be deterministic and thus repeatable and auditable. Publication of related code lists, for example either multiple versions of a single code list or other appropriate groupings, shall be accommodated to, for example, simplify configuration management tasks.

In any instance of a document that uses codes from a code list, it must be unambiguous what the set of valid codes are and the origin and version of the code list. For example, presuming that version can be facilitated by the definition of a unique Uniform Resource Identifiers (URI), it is required that:

1. Any two uses of the same namespace URI represent the use of the same code list definition

2. No two differing code list definitions shall be represented by the same namespace URI

3. When two trading partners identify the use of a code list, there must not be any ambiguity.

4. Should either partner create a code list or change an existing code list, the identification of the resulting code list must be distinct from that of its origin.

2.4.9 [R18 (Future)] Ability to prevent extension or modification

Certain code lists should not be extensible. For example, the traditional English list of colors in a rainbow, RED ORANGE YELLOW GREEN BLUE INDIGO VIOLET. It should be possible to indicate that such a code list is not extensible so the users can be assured of this constancy in its usage. [ABC] I think this only applies to XML Schema, not to the generic XML model, for reasons we can discuss as required.

2.5 Design Requirements of Code List Data Model

What follows is a list of some of the features that a code list data model must and/or should provide.

2.5.1 [R19] A set of the values (codes) forms each code list

Each code list must contain zero or more valid codes. The codes represent the content of the code list. Some useful code lists have been designed that have no specific predefined codes. Support for such lists is required.

2.5.2 [R20 (Future)] Multiple lists of equivalent values (codes) for a code list

Multiple representations for each code value must be supported in order to account for individual business requirements. For example, both integer & mnemonic representations may be needed as well as versions in more than one language. Clearly each value in a particular set of code values must be unique.

Consider days of the week, for instance. In this case well-accepted names, abbreviations, and integers might be needed under different circumstance to represent the text strings “Sunday”, “Monday”, “Tuesday”, etc. A number of different ways of identifying days are presented in the table below, and clearly the rightmost column is not acceptable since each value is no longer unique. This requirement means that there must be no impediments to adding columns to such a table of codes.

Number	Uppercase English	Mixed Case English	Mixed Case English Full	French Uppercase	Single Letter
0	SUN	Sun	Sunday	DIM	S
1	MON	Mon	Monday	LUN	M
2	TUE	Tue	Tuesday	MAR	T
3	WED	Wed	Wednesday	MER	W
4	THU	Thu	Thursday	JEU	T
5	FRI	Fri	Friday	VEN	F
6	SAT	Sat	Saturday	SAM	S

The format used to express each notional code or list entry should permit multiple values to be associated with or assigned to each such entry. List entries should be represented in a generic fashion that is appropriate both for all associated standards activities and for all conceivable code list implementations of the standard.

The format used for code lists should support any required level of complexity for both list entries and the code lists containing them. This format should make provision for the rapid construction of simple code lists and minimize the complexity of this process without increasing the difficulty of generating more complex lists. Any format should be portable and thus able to be processed on a broad range of computer systems.

2.5.3 [R21] Unique identifier(s) for a code list

Each code list and each version of such a list must contain at least one unique identifier (or set of identifiers which are collectively unique) able to reference that entire code list. It is equivalent to a key for the entire code list that can distinguish it from other code lists. There should be no restrictions as to which set of codes in the list can be used for this purpose, how many such keys will be used or which key(s) have higher priorities than others.

The unique identifier(s) for each code list shall support automated differentiation, i.e. by machine, of each code list or version thereof from all others.

2.5.4 [R22] Unique identifiers for individual entries in a code list

Each code within a code list must be represented by a unique identifier. This requirement means that no two codes within a single code list can have identical identifiers.

2.5.5 [R23] Names for a code list

Each code list must have a unique name. The same, as much as possible, should convey the content of the list.

2.5.6 [R24] Documentation for a code list

Each code list must contain documentation that describes, in detail, the business usage for that code list.

2.5.7 [R25] Documentation for individual entries on a code list

Each code entry on a code list shall support valid values, optional index values, and an optional long description to convey, in detail, the business meaning (as presented from the context of the code list author) and usage for this code value 

2.5.8 [R26 (Future)] The ability to import, extend, and/or restrict values and elements of other code lists

The model for code lists must provide the ability to extend, restrict or import additional values and/or elements of other code lists.

Each code list and the format used to represent it must support derivation of descendant code lists.

Derivation in this context shall include adding and/or removing notional codes and/or sets of values associated with the list as well as adding and/or removing keys, descriptive information, etc.

Any such derivation shall be done in a deterministic fashion that is repeatable and auditable (see [R17], [R21]).

2.5.9 [R27 (Future)] Support for describing code lists that cannot be enumerated

Provision shall be made for the creation of code lists that cannot be enumerated either in part or in their entirety because of size, volatility, or proprietary restrictions.

2.5.10 [R28 (Future)] Support for references to equivalent code lists

Each code list must be able to refer to other code lists that may or may not be used in place of it. These references are not necessarily exactly the same, but may be equivalent based on business usage.

If there are two code lists that can substitute for each other in a transaction, there shall be a mechanism by which this relationship can be expressed.

2.5.11 [R29 (Future)] Support for individual values to be mapped to equivalent values in other code lists

Each code list value must be able to refer to other code list values that may or may not be used in place of it. These references are not necessarily exactly the same, but may be equivalent based on business usage.

For example, a country might change its name, and hence be assigned a different country code, which is effectively a replacement for the previous one.

2.5.12 [R30 (Future)] Support for users to attach their own metadata to a code list

Each code list shall accommodate the addition of descriptive information by an individual user to account for unique business requirements.

Addition of such “metadata” to any combination of code lists, individual codes, and associated values shall be supported.

2.5.13 [R31 (Future)] Support for describing the validity period of the values

An effective date and expiration date should be established so that the code list can be scoped in time. See, for example, “Patterns for things that change with time”, http://martinfowler.com/ap2/timeNarrative.html.

2.5.14 [R32] Identifier for UN/CEFACT DE 3055.

Many code lists have been defined by UN/CEFACT. The code list model requires a representation of an identifier for this standard UNTDED 3055 **[UNTDED 3055]**. This identifier uniquely identifies UN/EDIFACT standard code lists.

3 Data and Metadata Model for Code Lists

This section provides rules for developing and using reusable code lists. These rules were developed for the UBL Library and derivations thereof, but they may also be used by other code-list-maintaining agencies as guidelines for any vocabulary wishing to share code lists. See section 5.0 Conformance.

Since the UBL Library is based on the ebXML Core Components Version 2.01, 15 November 2003; see **[CCTS2.01]**), the supplementary components identified for the *Code. Type* core component type are used to identify a code as being from a particular list.

Note that the model in this section is presented in two parts:

A data model for the codes themselves, and,

A metadata model for “supplementary components” that describe the entire list

3.1 Data Model Definition

The data model of codes in a code list is presented below.

CCT	UBL Name	Object Class	Property Term	Represen-tation Term	Primitive Type	Card.	Remarks
Code. Content	Content	Code	Content	Text	String	1..1	Required
Code. Name. Text	CodeName	Code	Name	Text	String	0..n	Optional
N/A	CodeDescription	Code Description	Description	Text	String	0..n	Optional
N/A	CodeIndex (Future)	Code Index	Index	Numeric	Number	0..1	Optional

3.2 Supplementary Components (Metadata) Model Definition

The following model contains the supplementary components description of a code list.

CCT	UBL Name	Object Class	Property Term	Represen-tation Term	Primitive Type	Card.	Remarks
N/A	name	Code	Name	Text	String	0..1	Optional
Code List. Identifier	CodeListID	Code List	Identification	Identifier	String	0..1	Optional
Code List. Agency. Identifier	CodeListAgencyID	Code List	Agency	Identifier	String	0..1	Optional

Code List. Agency Name. Text	CodeListAgencyName	Code List	Agency Name	Text	String	0..1	Optional
Code List. Name. Text	CodeListName	Code List	Name	Text	String	0..1	Optional
Code List. Version. Identifier	CodeListVersionID	Code List	Version	Identifier	String	0..1	Optional
Code List. Uniform Resource. Identifier	CodeListURI	Code List	Uniform Resource	Identifier	String	0..1	Optional
Code List Scheme. Uniform Resource. Identifier	CodeListSchemeURI	Code List Scheme	Uniform Resource	Identifier	String	0..1	Optional
Language. Identifier	LanguageID	Language	Identifier	Identifier	String	0..1	Optional
Code List . Namespace . Prefix. Identifier	CodeListNamespacePrefixID	Code List	Namespace Prefix	Identifier	String	0..1	Optional
N/A	CodeListDescription	Code List	Description	Text	String	0..1	Optional
N/A	CodeListCredits	Code List	Credits	Text	String	0..1	Optional

3.3 Examples of Use

The data type “Code“ is used for all elements that should enable coded value representation in the communication between partners or systems, in place of texts, methods, or characteristics. The list of codes should be relatively stable and should not be subject to frequent alterations (for example, CountryCode, LanguageCode, etc.). Code lists must have versions.

If the agency that manages the code list is not explicitly named and is specified using a role, then this takes place in an element type’s name.

The following types of code can be represented:

a.) Standardized codes whose code lists are managed by an agency from the code list DE 3055.

Code	Standard
CodeListID	Code list for standard code
CodeListVersionID	Code list version
CodeListAgencyID	Agency from DE 3055 (excluding roles)

b.) Proprietary codes whose code lists are managed by an agency that is identified by using a standard.

Code	Proprietary
CodeListID	Code list for the propriety code

CodeListVersionID	Version of the code list
CodeListAgencyID	Standardized ID for the agency (normally the company that manages the code list)
CodeListSchemeURI	ID schema for the schemeAgencyId
CodeListURI	Agency DE 3055 that manages the standardized ID ‘listAgencyId’

c.) Proprietary codes whose code lists are managed by an agency that is identified without the use of a standard.

Code	Proprietary
CodeListID	Code list for the proprietary code
CodeListVersionID	Code list version
CodeListAgencyID	Standardized ID for the agency (normally the company that manages the code list)
CodeListSchemeURI	ID schema for the schemeAgencyId
CodeListURI	‘ZZZ’ (mutually defined from DE 3055)

d.) Proprietary codes whose code lists are managed by an agency that is specified by using a role or that is not specified at all.

The role is specified as a prefix in the tag name. listID and listVersionID can optionally be used as attributes if there is more than one code list. If there is only one code list, no attributes are required.

Code	Proprietary
CodeListID	ID schema for the proprietary identifier
CodeListVersionID	ID schema version

4 XML Schema representation of Code Lists

[3/9/04 MJB] This section still needs correction to match the needs of the library content subcommittee when they settle on the specific set of supplementary components necessary when a code list is used as an element or as an attribute.

This section describes how the data model is mapped to XML schema **[XSD]**. The code list mechanism described in this paper assumes that it will be used in the UBL context according to the following graphic that describes the type derivation hierarchy for code list and related schemas **[UBL1-SD]**:

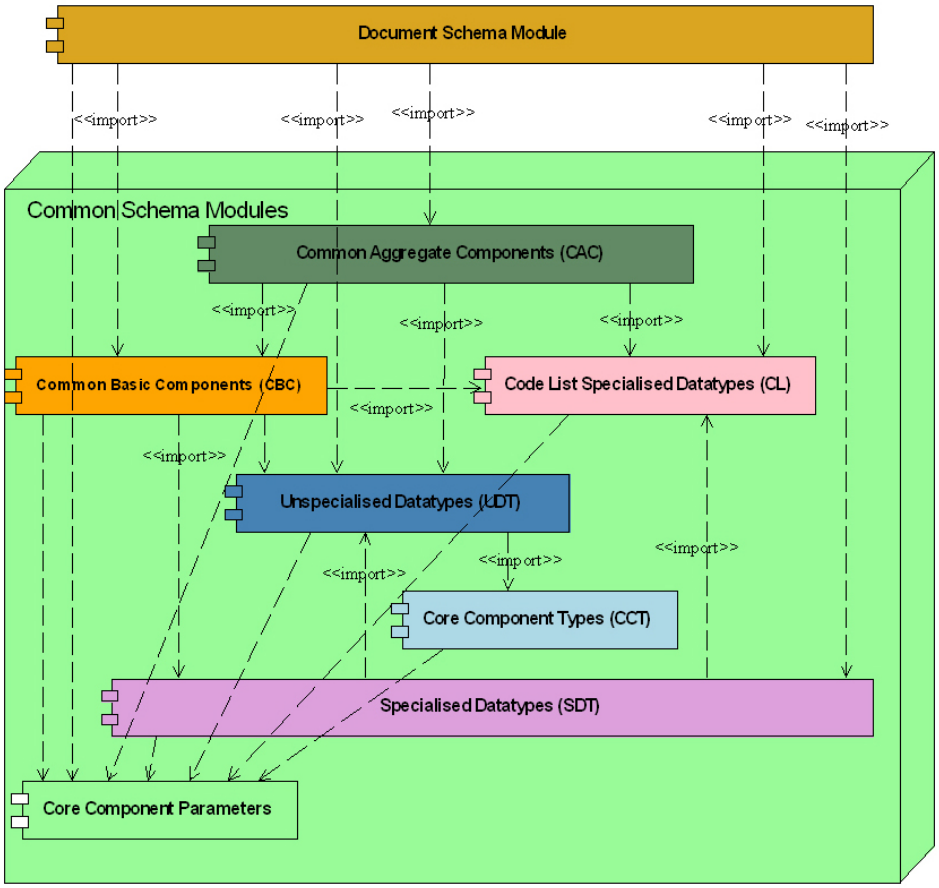

Figure 1 UML Diagram of UBL Schemas type hierarchy

As shown in the figure, an abstract model of "any" UBL code list appears in a code list specific namespace.

Note that an instance of a code list is derived in several pieces – a simpleType that contains the actual content of the code list, and, a complexType with simple content that attaches the optional supplementary components to the enumeration. The following procedure describes the construction of a code list schema:

- Define an abstract element for inclusion in extensible schemas (future)
- Define a simpleType to hold the enumerated values
- Define a complexType to add the supplementary components

- Define a global attribute to contain the enumerated values as an attribute and for supplementary components as needed. (future)
- Define an element that substitutes for the abstract type to enable usage in unextended schemas (future)
- Define a comprehensive URN to hold supplementary components that can qualify uniqueness of usage (future)

4.1 Data Model Mapping

The following table summarizes the component mapping of the data model. Items in braces, “{}” are references to the data model components. For example:

{code.name} represents the contents of the name of the code list, i.e. CountryCode;

“{code.name} Type” represents the contents of the name of the code list, i.e. “CountryCodeType”;

UBL Name	XMLSchema Mapping

o Code.Content	o 1. Abstract element (Future) `<xs:element name="{code.name}A" type="xs:token" abstract="true"/>` o 2. Simple type to hold code list values and optional annotations `<xs:simpleType name="{code.name}Type"><xs:restriction base="xs:token"><xs:enumeration value="{code.content}"><xs:annotation><xs:documentation>{code.description}</xs:documentation></xs:annotation></xs:enumeration><xs:enumeration value="{code.content}"/><xs:enumeration value="{code.content}"/>. . .</xs:restriction></xs:simpleType>` o 3. Complex type to associate supplementary values with code list values that substitutes for the abstract type. `<xs:complexType name="{code.name}"><xs:annotation><xs:documentation><ccts:Instance><!-- Data and values stored in this space are meant for instance-processing purposes, and are non-normative. --><ccts:Prefix>loc</ccts:Prefix><ccts:CodeListQualifier>{code.name}</ccts:CodeListQualifier><ccts:CodeListAgency>{Code.listAgencyID}</ccts:CodeListAgency><ccts:CodeListVersion>{Code.listVersionID}</ccts:CodeListVersion></ccts:Instance></xs:documentation></xs:annotation><xs:simpleContent><xs:extension base="{Code.name}Type"><xs:attribute name="CodeListID" type="xs:token" fixed="{CodeListID}"/><xs:attribute name="CodeListAgencyID" type="xs:token" fixed="{CodeListAgencyID}"/><xs:attribute name="CodeListVersionID" type="xs:string" fixed="{CodeListVersionID}"/>. . . additional optional attributes</xs:extension></xs:simpleContent></xs:complexType>` o 4. Attribute (Future) `<xs:attribute name="{Code.name}" type="{Code.name}ContentType"/>` o 5. Element to substitute for abstract element in non-exended schemas (Future) `<xs:element name="{Code.name}" type="{Code.name}Type" substitutionGroup="{Code.name}TypeA"/>`

o Code.Description	Xs:annotation/ xs:documentation/
o Code.Value	Xs:annotation/ xs:documentation/

4.2 Supplementary Components Mapping

The following table shows all supplementary components of the code type. It also shows the current representation by using attributes and the recommended optional representation by using namespaces and annotations.

UBL Name	Optional XMLSchema Mapping URN mapping	Optional complex type attribute mapping
name	xs:annotation/ xs:documentation/ cc:codename	o This is the default name of the implemented element and attribute above.
CodeListID	namespace (URN) 1. position Mandatory	`<xs:attribute name="CodeListID" type="xs:normalizedString"/>`
CodeListName	namespace (URN) 2. position Optional	`<xs:attribute name="CodeListName" type="xs:string"/>`
CodeListVersionID	namespace (URN) 3. position Mandatory	`<xs:attribute name="CodeListVersionID" type="xs:normalizedString"/>`
CodeListAgencyID	namespace (URN) 4. position Optional	`<xs:attribute name="CodeListAgencyID" type="xs:normalizedString"/>`
CodeListAgencyName	namespace (URN) 5. position optional	`<xs:attribute name="CodeListAgencyName" type="xs:string"/>`
CodeListURI	namespace (URN) 6. position optional	`<xs:attribute name="CodeListURI " type="xs:anyURI"/>`
CodeListSchemeURI	namespace (URN) 7. position optional	`<xs:attribute name=" CodeListSchemeURI " type="xs:normalizedString"/>`
LanguageID		`<xs:attribute name=”LanguageID” type=”xs:language”/>`
CodeListNamespacePrefixID		`<xs:attribute name=” CodeListNamespacePrefixID” type=”xs:normalizedString”/>`
CodeListDescription		`<xs:attribute name=” CodeListDescription” type=”xs:string”/>`

CodeListCredits		`<xs:attribute name=" CodeListCredits" type="xs:string"/>`

4.3 Namespace URN (Future)

The following construct represents the construct for the URN of a code list, according OASIS URN:

```
urn:oasis:tc:ubl:codeList:<CodeList.Identification.Identifier>:<CodeList.Name.
Text>:<CodeList.Version.Identifier>:<CodeList.AgencyIdentifier>:<CodeList.Agen
cyName.Text>:<CodeList.AgencyScheme.Identifier>:<CodeList.AgencySchemeAgency.I
dentifier>
```

The first four parameters are fixed by Uniform Resource Name (URN) [see RFC 2141] and OASIS URN [see RFC 3121]:

- urn --> leading token of URNs
- oasis --> registered namespace ID “oasis”
- tc --> Technical Committee Work Products
- ubl --> From Technical Committee UBL (Universal Business Language)
- The parameter “codeList” identifies the schema type “code list”.
- The following parameters from <Code List. Identifier> to <Code List. Agency Scheme Agency. Identifier> represents the specific code list supplementary components of the CCT codeType.
- Example:

```
urn:oasis:tc:ubl:codeList:ISO639:Language%20Code:3:ISO:International%20Standar
dization%20Organization::
```

4.4 Namespace Prefix

REWORD THIS. Namespace prefix could be freely defined. However, it is helpful for better understanding, to identity the code lists by a convention of namespace prefixes.

The prefix provides the namespace prefix part of the qualified name of each code list. It is recommended that this prefix should contain the information of the supplementary component <Code List. Identification Identifier> and if it is necessary for separation, the information of the supplementary component <Code List. Version. Identifier> separated by a dash “-“. All letters should be lower case.

Example:

```
iso639
iso639-3 (with version)
```

4.5 Code List Schema Generation

This section describes how to generate complete code list schemas from the data model of section 4.

4.5.1 Data model and example values

The code list model and supplementary components are listed in the following table. The first column contains the UBL name and the second column contains an example of the value(s) for that name. It is assumed that the UBL name is the proposed name for the schema element/attribute/simpleType/complexType etc….

The expressions ValueOf(<UBL Name>), and, {UBL Name}refer to the contents for a specific code list. The latter representation is used so that a substitution can be shown within the schema fragments generated.

UBL Name	Description	Sample ValueOf(<UBL Name>) ≡ {UBL Name}
Content	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*.	<enumerated values>
Name	<enumerated value definitions> (if Content="USD" then Name = "US Dollars")	The textual name of the code content.
CodeListID	The identification of a list of codes.	ISO4217 Alpha
CodeListAgencyID	An agency that maintains one or more code lists.	6
CodeListAgencyName	The name of the agency that maintains the code list.	United Nations Economic Commission for Europe
CodeListName	The name of a list of codes.	Currency
CodeListVersionID	The *Version* of the code list.	0.3
CodeListURI	The Uniform Resource Identifier that identifies where the code list is located.	http://www.bsi-global.com/Technical%2BInformation/Publications/_Publications/tig90x.doc
CodeListSchemeURI	The Uniform Resource Identifier that identifies where the code list scheme is located.	urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0-draft-8-11
LanguageID	The identifier of the language used in the corresponding text string	En
CodeListNamespacePrefixID	The namespace prefix recommended for this code list. Should be based on the CodeListID.	cur
CodeListDescription	Describes the set of codes	The set of world currencies
CodeListCredits	Acknowledges the source and ownership of codes	Derived from the ISO 4217 currency code list and used under the terms of the ISO policy stated at http://www.iso.org/iso/en/commcentre/pressreleases/2003/Ref871.html.

4.5.2 Schema to generate

This section describes the specific steps required to generate a schema from the above model. Each step shows two schema fragments – one that is a template for generating the schema, and, the second one

that is an example schema generated. In the template sections, the places where values from the spreadsheet model are inserted are shown in braces, and are colored green –

e.g. “{CodeListAgencyID}” means substitute the value “6”.

4.5.3 Schema file name

The name of this schema file should be:

UBL-CodeList-{CodeListName}-{CodeListVersionID}.xsd

For example:

UBL-CodeList-CurrencyCode-1.0.xsd

4.5.3.1 Generate XML header

Template, Sample are the same:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Universal Business Language (UBL) Schema 1.0-draft-10.1

  Copyright (C) OASIS Open (2004). All Rights Reserved.

  This document and translations of it may be copied and furnished to others, and
  derivative works that comment on or otherwise explain it or assist in its
  implementation may be prepared, copied, published and distributed, in whole or
  in part, without restriction of any kind, provided that the above copyright
  notice and this paragraph are included on all such copies and derivative works.
  However, this document itself may not be modified in any way, such as by
  removing the copyright notice or references to OASIS, except as needed for the
  purpose of developing OASIS specifications, in which case the procedures for
  copyrights defined in the OASIS Intellectual Property Rights document must be
  followed, or as required to translate it into languages other than English.

  The limited permissions granted above are perpetual and will not be revoked by
  OASIS or its successors or assigns.

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

  ============================================================================

  For our absent friend, Michael J. Adcock  - il miglior fabbro

  ============================================================================

  Universal Business Language Specification
     (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl)
  OASIS Open (http://www.oasis-open.org/)

  Schema generated by GEFEG EDIFIX v5.0-beta
     (http://www.gefeg.com/en/standard/xml/ubl.htm)


  Document Type:     CurrencyCode
  Generated On:      Fri Mar 26 14:30:20 2004
-->
```

4.5.3.2 Generate XML Schema header

Template:

```
<xs:schema
    targetNamespace="{CodeListSchemeURI}"
    xmlns="{CodeListSchemeURI}"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified" version="1:0-draft-7.1">
```

562 **Sample:**

```
<xs:schema
    targetNamespace="urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0-draft-7.1"
                xmlns="urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0-draft-7.1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified" version="1:0-draft-7.1">
```

563 #### 4.5.3.3 Generate abstract element (Future)

564 **Template:**

```
<xs:element name="{CodeListName}Abstract" type="xs:string" abstract="true"/>  {i would prefer to make the meaning of this clear}
```

565 **Sample:**

```
<xs:element name="CurrencyCodeAbstract" type="xs:normalizedString" abstract="true"/>
```

566 #### 4.5.3.4 Generate simple type to contain the enumerated values

567 **Template:**

```
    <xs:simpleType name="{CodeListName}ContentType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="{first Content}"
                <xs:annotation>
                    <xs:documentation>
                        <CodeName>{first Name}"</CodeName>
                    </xs:documentation>
                </xs:annotation>
            </xs:enumeration>
                    …
            <xs:enumeration value="{last Content}"
                 <xs:annotation>
                    <xs:documentation>
                        <CodeName>{last  Name}"</CodeName>
                    </xs:documentation>
                </xs:annotation>
             </xs:enumeration>
        </xs:restriction>
    </xs:simpleType>
```

568 **Sample:**

```
<xs:simpleType name="CurrencyCodeContentType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="AED">
            <xs:annotation>
                <xs:documentation>
                    <CodeName>UAE Dirham</CodeName>
                </xs:documentation>
            </xs:annotation>
        </ xs:enumeration>
        <xs:enumeration value="ALL">
            <xs:annotation>
                <xs:documentation>
                    <CodeName>Albanian Lek</CodeName>
                </xs:documentation>
            </xs:annotation>
        </xs:xs:enumeration>
        <xs:enumeration value="AMD"
            <xs:annotation>
                <xs:documentation>
                    <CodeName>Armenian Dram</CodeName>
                </xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="ANG"/>
        <xs:enumeration value="AOA"/>
        <xs:enumeration value="XDR"/>
                …
        <xs:enumeration value="ZAR"/>
        <xs:enumeration value="ZMK"/>
        <xs:enumeration value="ZWD"/>
    </xs:restriction>
</xs:simpleType>
```

4.5.3.5 Generate complex type to hold enumerated values and supplemental components

Template:

```
<xs:complexType name="{CodeListName}Type">
    <xs:annotation>
        <xsd:documentation>
            <ccts:Component>
                <ccts:ComponentType>DT</ccts:ComponentType>
                <ccts:DictionaryEntryName>Code. Type</ccts:DictionaryEntryName>
                <ccts:RepresentationTerm>Code</ccts:RepresentationTerm>
                <ccts:DataTypeQualifier>Currency</ccts:DataTypeQualifier>
                <ccts:DataType>Code. Type</ccts:DataType>
            </ccts:Component>
            <ccts:Instance>
                <ccts:CodeListID>{CodeListID}</ccts:CodeListID>
                <ccts:CodeListAgencyID>{CodeListAgencyID}</ccts:CodeListAgencyID>
                <ccts:CodeListAgencyName>{CodeListAgencyName}</ccts:CodeListAgencyName>
                <ccts:CodeListName>{CodeListName}</ccts:CodeListName>
                <ccts:CodeListVersionID>{CodeListVersionID}</ccts:CodeListVersionID>
                <ccts:CodeListUniformResourceID>{CodeListURI}</ccts:CodeListUniformResourceID>
                <ccts:CodeListSchemeUniformResourceID>{CodeListSchemeURI}
                        </ccts:CodeListSchemeUniformResourceID>
                <ccts:LanguageID>{LanguageID}</ccts:LanguageID>
            </ccts:Instance>
        </xsd:documentation>
    </xs:annotation>
    <xs:simpleContent>
        <xs:extension base="{CodeListName}ContentType">
            <xs:attribute name="name" type="xs:string" use="optional"/>  ?????????
            <xs:attribute name="codeListID" type="xs:normalizedString" fixed="{CodeListID}"/>
            <xs:attribute name="codeListAgencyID" type="xs:normalizedString"
                fixed="{CodeListAgencyID}"/>
            <xs:attribute name="codeListAgencyName" type="xs:normalizedString"
                fixed="{CodeListAgencyName}"/>
            <xs:attribute name="codeListName" type="xs:string" fixed="{CodeListName}">
            <xs:attribute name="codeListVersionID" type="xs:string"
                fixed="{CodeListVersionID}"/>
            <xs:attribute name="codeListURI" type="xs:anyURI" fixed="{CodeListURI}">
            <xs:attribute name="codeListSchemeURI" type="xs:anyURI"
                fixed="{CodeListSchemeURI}">
            <xs:attribute name="languageID" type="xs:language" fixed="{LanguageID}">
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

572 **Sample:**

```
    <xs:complexType name="CurrencyCodeType">
        <xs:annotation>
            <xsd:documentation>
                <ccts:Component>
                    <ccts:ComponentType>DT</ccts:ComponentType>
                    <ccts:DictionaryEntryName>Code. Type</ccts:DictionaryEntryName>
                    <ccts:RepresentationTerm>Code</ccts:RepresentationTerm>
                    <ccts:DataTypeQualifier>Currency</ccts:DataTypeQualifier>
                    <ccts:DataType>Code. Type</ccts:DataType>
                </ccts:Component>
                <ccts:Instance>
                    <ccts:CodeListID>ISO 4217 Alpha</ccts:CodeListID>
                    <ccts:CodeListAgencyID>6</ccts:CodeListAgencyID>
                    <ccts:CodeListAgencyName>United Nations Economic Commission for
Europe</ccts:CodeListAgencyName>
                    <ccts:CodeListName>Currency</ccts:CodeListName>
                    <ccts:CodeListVersionID>0.3</ccts:CodeListVersionID>
                    <ccts:CodeListUniformResourceID>
                        http://www.bsi-global.com/Technical%2BInformation
                        /Publications/_Publications/tig90x.doc </ccts:CodeListUniformResourceID>
                    <ccts:CodeListSchemeUniformResourceID>
                        urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0-draft-10.1
                        </ccts:CodeListSchemeUniformResourceID>
                    <ccts:LanguageID>en</ccts:LanguageID>
                </ccts:Instance>
            </xsd:documentation>
        </xs:annotation>
        <xs:simpleContent>
            <xs:extension base="CurrencyCodeContentType">
                <xsd:attribute name="name" type="xsd:string" use="optional"/>
                <xsd:attribute name="codeListID" type="xsd:normalizedString" use="optional"
                        fixed="ISO 4217 Alpha"/>
                <xsd:attribute name="codeListAgencyID" type="xsd:normalizedString" use="optional"
                        fixed="6"/>
                <xsd:attribute name="codeListAgencyName" type="xsd:string" use="optional"
                        fixed="United Nations Economic Commission for Europe"/>
                <xsd:attribute name="codeListName" type="xsd:string" use="optional"
                        fixed="Currency"/>
                <xsd:attribute name="codeListVersionID" type="xsd:normalizedString" use="optional"
                        fixed="0.3"/>
                <xsd:attribute name="codeListURI" type="xsd:anyURI" use="optional"
                    fixed="http://www.bsi-global.com/
                        Technical%2BInformation/Publications/_Publications/tig90x.doc"/>
                <xsd:attribute name="codeListSchemeURI" type="xsd:anyURI" use="optional"
                    fixed="urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0-draft-10.1"/>
                <xsd:attribute name="languageID" type="xsd:language" use="optional" fixed="en"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

4.5.3.6 Generate global attributes to allow usage of code lists as an attribute (Future)

Template:

```
<xs:attribute name="{CodeListName}" type="{CodeListName}ContentType"/>
<xs:attribute name="codeListID" type="xs:normalizedString" fixed="{CodeListID}"/>
<xs:attribute name="codeListAgencyID" type="xs:normalizedString " fixed="{CodeListAgencyID}"/>
<xs:attribute name="codeListAgencyName" type="xs:string"
        fixed="{CodeListAgencyName}"/>
<xs:attribute name="codeListVersionID" type="xs:normalizedString " fixed="{CodeListVersionID}"/>
<xs:attribute name="codeListName" type="xs:string " fixed="{CodeListName}"/>
<xs:attribute name="name" type="xs:normalizedString " fixed="{name}"/>
<xs:attribute name="codeListURI" type="xs:anyURI" fixed="{CodeListURI}"/>
<xs:attribute name="codeListSchemeURI" type="xs:anyURI" fixed="{CodeListSchemeURI}"/>
<xs:attribute name="languageID" type="xs:normalizedString " fixed="{LanguageID}"/>
```

Sample:

```
<xs:attribute name="CurrencyCode" type="CurrencyCodeContentType"/>
<xs:attribute name="name" type="xs:normalizedString" fixed="cur"/>
<xs:attribute name="codeListID" type="xs:normalizedString" fixed="ISO 4217 Alpha"/>
<xs:attribute name="codeListAgencyID" type="xs:normalizedString " fixed="6"/>
<xs:attribute name="codeListAgencyName" type="xs:string "
        fixed="United Nations Economic Commission for Europe"/>
<xs:attribute name="codeListVersionID" type="xs:normalizedString " fixed="0.3"/>
<xs:attribute name="codeListName" type="xs:string" fixed="CurrencyCode"/>
<xs:attribute name="codeListURI" type="xs:anyURI"
    fixed="http://www.bsi-global.com/Technical%2BInformation/Publications/_Publications/tig90x.doc"/>
<xs:attribute name="codeListSchemeURI" type="xs:anyURI"
     fixed="urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0-draft-8-1"/>
<xs:attribute name="languageID" type="xs:language" fixed="en"/>
```

4.5.3.7 Generate global element to allow usage of code list as an element (Future)

Template:

```
<xs:element name="{CodeListName}" type="{CodeListName}Type"
substitutionGroup="{CodeListName}Abstract"/>
```

Sample:

```
<xs:element name="CurrencyCode" type="CurrencyCodeType"
        substitutionGroup="CurrencyCodeAbstract"/>
```

4.5.3.8 End of schema

Template:

```
</xs:schema>
```

Sample:

```
</xs:schema>
```

4.6 Code List Schema Usage

For every code list, there exists a specific code list schema. This code list schema must have a targetNamespace with the UBL specific code list namespace and have a prefix with the code list identifier itself.

The element in the code list schema can be used for the representation as a global declared element in the document schemas. The name of the element is the UBL tag name of the specific BIE for a code.

The simpleType represents the possible codes and the characteristics of the code content. The name of the simpleType must be always ended with ". Content". Within the simpleType is a restriction of the XSD built-in data type "xs:token". This restriction includes the specific facets "length", "minLength", "maxLength" and "pattern" for regular expressions to describe the specific characteristics of each code list.

Each code will be represented by the facet “enumeration” after the characteristics. The value of each enumeration represents the specific code value and the annotation includes the further definition of each code, like “Code. Name”, “Language. Identifier” and the description.

The schema definitions to support this might look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
   targetNamespace="urn:oasis:ubl:codeList:ISO3166:Locale%20Code:3:5:ISO::"
   xmlns:iso3166="urn:oasis:ubl:codeList:ISO3166: Locale%20Code:3:5:ISO::"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified">

<xs:element name="LocaleCodeTypeA" type="xs:token"
    abstract="true">
  <xs:annotation>
    <xs:documentation>
   An abstract place holder for a code list element
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:simpleType name="LocaleCodeContentType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="DE"/>
    <xs:enumeration value="FR"/>
    <xs:enumeration value="US"/>
    . . .
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="LocaleCodeType">
   <xs:annotation>
      <xs:documentation>
        <ccts:Instance>
          <!-- Data and values stored in this space
          are meant for instance-processing purposes, and are
          non-normative. -->
          <ccts:Prefix>loc</ccts:Prefix>
          <ccts:CodeListQualifier>LocaleCode</ccts:CodeListQualifier>
          <ccts:CodeListAgency>ISO3166</ccts:CodeListAgency>
          <ccts:CodeListVersion>0.3</ccts:CodeListVersion>
        </ccts:Instance>
      </xs:documentation>
   </xs:annotation>
   <xs:simpleContent>
      <xs:extension base=" LocaleCodeType">
         <xs:attribute name="CodeListID" type="xs:token" fixed="ISO3166"/>
         <xs:attribute name="CodeListAgencyID" type="xs:token" fixed="6"/>
         <xs:attribute name="CodeListVersionID" type="xs:string" fixed="0.3"/>
            . . . additional optional attributes
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>

<xs:element name="LocaleCode" type="LocaleCodeType"
               substitutionGroup="LocaleCodeTypeA">
    <xs:annotation>
      <xs:documentation>
         A substitution for the abstract element based
         on aStdEnum
      </xs:documentation>
    </xs:annotation>
</xs:element>
```

```
<xs:attribute name="{Code.name}" type="{Code.name}ContentType">
    <xs:annotation>
      <xs:documentation>
          A global attribute for use inside an element
      </xs:documentation>
    </xs:annotation>
< xs:attribute/>


</xs:schema>
```

4.7 Instance

The enumerated list method results in instance documents with the following structures.

```
<LocaleCode>US</LocaleCode>

<iso3166:LocaleCode>US</iso3166:LocaleCode>

<PostCode iso3166:LocaleCode="FQ">20878</PostCode>
```

4.8 Deriving New Code Lists from Old Ones (future)

In order to promote maximum reusability and ease code lists maintenance, code list designers are expected to build new code lists from existing lists. They could for example combine several code lists or restrict an existing code list.

These new code lists must be usable in UBL elements the same manner the “basic” code lists are used.

4.8.1 Extending code lists

The base schema shown above could be extended to support new codes as follows:

```
<xs:schema targetNamespace="cust"
  xmlns:std="std"
  xmlns="cust"
  xmlns:cust="custom"
  xmlns:xs=http://www.w3.org/2001/XMLSchema
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

<xs:import namespace="std"
  schemaLocation="D:\_PROJECT\NIST\XMLSchema\test0513\std.xsd"/>

<xs:element name="LocaleCode" substitutionGroup="std:LocaleCodeA">
  <xs:annotation>
    <xs:documentation>A substitute for the abstract LocaleCodeA
      that extends the enumeration
    </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:union memberTypes="std:aStdEnum">
      <xs:simpleType>
        <xs:restriction base="xs:token">
         <xs:enumeration value="IL"/>
         <xs:enumeration value="GR"/>
        </xs:restriction>
      </xs:simpleType>
```

```
    </xs:union>
  </xs:simpleType>
</xs:element>
</xs:schema>
```

4.8.2 Restricting code lists

The base schema shown above could be restricted to support a subset of codes as follows:

```
<xs:import namespace="std"
  schemaLocation="D:\_PROJECT\NIST\XMLSchema\test0513\std.xsd"/>
<xs:element name="LocaleCode" substitutionGroup="std:LocaleCodeA">
  <xs:annotation>
    <xs:documentation>
      A substitute for the abstract LocaleCodeA that restricts
        the enumeration
    </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:token">
    <xs:enumeration value="DE"/>
    <xs:enumeration value="US"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

5 Conformance to UBL Code Lists (future)

This section is for Producers of Code Lists outside of UBL. These lists could be owned by a number of different types of organizations.

We probably need a Conformance section in this document so that code list producers (who, in general, won't be UBL itself) will know how/when to claim conformance to the requirements (MUST) and recommendations (SHOULD/MAY) in this specification. This spec is not for the UBL TC, but for code list producers (which may occasionally include UBL itself).

6 References

[3166-XSD] UN/ECE XSD code list module for ISO 3166-1,

[CCTS2.01] *UN/CEFACT Core Components Technical Specification – Part 8 of the ebXML Framework*, 15 November 2003, Version 2.01.

[CLSC] OASIS UBL Code List Subcommittee. Portal: http://www.oasis-open.org/committees/sc_home.php?wg_abbrev=ubl-clsc . Email archive: http://lists.oasis-open.org/archives/ubl-clsc/.

[SPENCER] http://www.oasis-open.org/apps/org/workgroup/ubl-clsc/download.php/5195/Spencer-CodeList-PositionPaper1-0.pdf

[STUHEC] <need reference>

[COATES] **http://www.oasis-open.org/apps/org/workgroup/ubl-clsc/download.php/4522/draft-coates-codeListDataModels-0p2.doc**

[CLTemplate] OASIS UBL Naming and Design Rules code list module template, http://www.oasis-open.org/committees/ubl/ndrsc/archive/.

[eBSC] "eBusiness Standards Convergence Forum", http://www.nist.gov/ebsc.

[eBSCMemo] M. Burns, S. Damodaran, F.Yang, "Draft Code List Implementation description", http://www.oasis-open.org/apps/org/workgroup/ubl-clsc/download.php/4503/nistTOUbl20031119.zip

[NDR] M. Cournane et al., *Universal Business Language (UBL) Naming and Design Rules*, OASIS, 2002, http://www.oasis-open.org/committees/ubl/ndrsc/archive/wd-ublndrsc-ndrdoc-nn/.

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[CL5] **http://www.oasis-open.org/apps/org/workgroup/ubl-clsc/download.php/4502/wd-ublndrsc-codelist-05_las_20030702.doc**

[ISO 11179]

http://www.iso.org/iso/en/StandardsQueryFormHandler.StandardsQueryFormHandler?scope=CATALOGUE&keyword=&isoNumber=11179

[UBL1-SD] http://ibiblio.org/bosak/ubl/UBL-1.0/art/UBL-1.0-SchemaDependency.jpg

[UNTDED 3055] <need reference>

[XSD] *XML Schema*, W3C Recommendations Parts 0, 1, and 2. 2 May 2001. http://www.unece.org/etrades/unedocs/repository/codelist.htm.

Appendix A. Revision History

Revision	Editor	Description
2004-01-13	Marty Burns	First complete version converted from NDR revision 05
2004-01-14	Marty Burns	Minor edit of chapter heading 3 & 4
2004-01-20	Marty Burns	Incorporated descriptions from AS and KH
2004-02-06	Marty Burns	Cleaned up requirements and other sections – removed some redundant content from merge of contributions. Explicitly identified Data Model and Metadata models separately from XML representations of the same.
2004-02-11	Marty Burns	Added comments from 2/11 conference call
2004-02-29	Marty Burns	Added resolutions from February Face to Face meeting
2004-03-03	Marty Burns	Incorporated Tim McGrath's corrections of data model
2004-03-09	Marty Burns	Addressed Eve Maler's comments Addressed Tony Coates comments Addressed 2004-03-03 telecon comments Added some elaboration of the model usage in ubl
2004-03-15	Marty Burns	Added example mapping schema paper to section 4.6
2004-03-23	Marty Burns	Added data model for supplementary components, Marked future features for UBL 1.1 as (future) Added comment about UBL1.0 release vs. future.
2004-04-01	Marty Burns	Clean up for UBL version 1.0
2004-04-14	Marty Burns	Incorporated suggested edits from GKH
2005-01-02	Marty Burns	Incorporated elaborations of requirements for better clarity to kick off the UBL 1.1 revisions. Incorporated comments from Tony Coates.

Appendix B. Notices

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

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