



# Practical Code List Implementation (Using Controlled Vocabularies in XML Documents)

Crane Softwrights Ltd.  
<http://www.CraneSoftwrights.com>





# Practical Code List Implementation (Using Controlled Vocabularies in XML Documents)

Crane Softwrights Ltd.  
<http://www.CraneSoftwrights.com>

## Copyrights

- Other original material herein is copyright (C) 1998-2009 Crane Softwrights Ltd. This is commercial material and may not be copied or distributed by any means whatsoever without the expressed permission of Crane Softwrights Ltd.

## Disclaimer

- By purchasing and/or using any product from Crane Softwrights Ltd. ("Crane"), the product user ("reader") understands that this product may contain errors and/or other inaccuracies that may result in a failure to use the product itself or other software claiming to utilize any proposed or finalized standards or recommendations referenced therein. Consequently, it is provided "AS IS" and Crane disclaims any warranty, conditions, or liability obligations to the reader of any kind. The reader understands and agrees that Crane does not make any express, implied, or statutory warranty or condition of any kind for the product including, but not limited to, any warranty or condition with regard to satisfactory quality, merchantable quality, merchantability or fitness for any particular purpose, or such arising by law, statute, usage of trade, course of dealing or otherwise. In no event will Crane be liable for (a) punitive or aggravated damages; (b) any direct or indirect damages, including any lost profits, lost savings, damaged data or other commercial or economic loss, or any other incidental or consequential damages even if Crane or any of its representatives have been advised of the possibility of such damages or they are foreseeable; or (c) for any claim of any kind by any other party. Reader acknowledges and agrees that they bear the entire risk as to the quality of the product.

## OASIS Copyrights

- Some information included in this publication is from copyrighted material from the Organization for the Advancement of Structured Information Standards (OASIS) <http://www.oasis-open.org>. Files containing the copyrighted material include the following, which applies only to original OASIS documents and not to this commercial material created by Crane; please go to original OASIS documents to obtain any publicly-available content:
  - Portions copyright (C) OASIS Open 2001-2009. All Rights Reserved.
  - <http://www.oasis-open.org/who/intellectualproperty.php>

# Practical Code List Implementation (Prelude) (cont.)



## Preface

The main content of this book is in an unconventional style primarily in bulleted form

- derivations of the book are used for instructor-led training, requiring the succinct presentation
  - note the exercises included in instructor-led training sessions are not included in the book
- derivations of the book can be licensed and branded for customer use in delivering training
- the objective of this style is to convey the essence and details desired in a compact, easily perused form, thereby reducing the search for key words and phrases in lengthy paragraphs
- each chapter of the book corresponds to a module of the training
- each page of the book corresponds to a frame presented in the training
- a summary of subsections and their pages is at the back of the book

Much of the content is hyperlinked both internally and externally to the book in the 1-up full-page sized electronic renditions:

- (note the Acrobat Reader "back" keystroke sequence is "Ctrl-Left")
- page references (e.g.: OASIS Genericcode 1.0 (page 55))
- external references (e.g.: <http://docs.oasis-open.org/codelist/genericcode>)
- chapter references in book summary
- section references in chapter summary
- subsection references in table of contents at the back of the book
- hyperlinks are not present in the cut, stacked, half-page, or 2-up renditions of the material

## Diagram legend:

- triangle: a structured XML/SGML/HTML document or resource
- parallelogram: a non-structured document or resource in an arbitrary format
- box: a process in the work flow
- diamond: a decision in the workflow

## Sample code fragments:

- included with the book purchase is a ZIP file of sample code fragments
- directory names referenced in the book are referencing subdirectories in the unpacked ZIP files
- the `readme.txt` file in the ZIP package documents the running of sample batch files and shell scripts

# Practical Code List Implementation (Prelude) (cont.)



## Important caveat regarding the information in this publication

- while the author, G. Ken Holman, is the chairman of the OASIS Code List Representation Technical Committee, not all of the material in this publication is necessarily accepted by all of the TC members as some of it is just proposed for acceptance
- all of the content in this book is written from the opinion of G. Ken Holman and Crane Softwrights Ltd. and does not necessarily represent official or agreed-upon content from the perspective of the CLRTC
- this content is not to be construed as legal advice of any kind, nor is it recommending that any particular methodology or process or tool be implemented, it only documents methodologies and technologies available to be considered

Entire chapters of this publication will undergo revision

- the CLRTC is debating the CVA specification
- some software being developed by Crane Softwrights Ltd. is being made freely available for anyone to download and use
- some software being developed by Crane Softwrights Ltd. will be made available only to customers of this publication
  - to supplement the software that is made freely available

The purchase of this publication is protected by the no-charge availability of all future editions

- all of the content in this publication is subject to revision and update and editions will get out of date
- early editions are expected to be created frequently and be short-lived as the community experience with the code list file formats and processes reveals various practices and experiences that will influence how to consider working with this standard

The purchase of this publication grants the legitimate owner no-charge access to accompanying software written by Crane Softwrights Ltd.

- there are no warranties expressed or implied regarding the use of the software; more details are found in the documentation for the software
- access details to download the software are found by registered users on <http://www.CraneSoftwrights.com/sales/pcli/>
- while the software is free of charge, the software is not to be copied for or distributed to or used by anyone who is not a legitimate customer of this book, unless permission has been granted in writing

The author welcomes any and all suggestions for improvements and additional content

- please do not hesitate to contribute ideas for improving on this publication
- all submissions to [info@CraneSoftwrights.com](mailto:info@CraneSoftwrights.com) will be acknowledged (though not necessarily accepted!)
  - please note that aggressive spam filters may make our email delivery difficult

# Practical Code List Implementation



- Introduction - Practical Code List Implementation
- Chapter 1 - Controlled vocabularies
- Chapter 2 - Defining and using controlled vocabularies
- Chapter 3 - Declaring controlled vocabularies
- Chapter 4 - Controlled vocabulary representation detail
- Chapter 5 - Associating controlled vocabularies in XML documents
- Chapter 6 - Controlled vocabulary association detail
- Chapter 7 - Your own business document controlled vocabulary
- Annex A - OpenOffice 3 genericcode and CVA filters
- Conclusion - Where to go from here?

Series: Practical Code List Implementation

Reference: Electronic commerce

Outcome

- detailed review of the concepts, implementation and deployment of code lists

# Practical Code List Implementation

Introduction - Practical Code List Implementation



This book is oriented to the system architect and system developer interested in deploying OASIS specifications for code lists in XML

- how do external code lists differ from embedded code lists and to what benefit?
- what information design facets support flexible code list deployment?
- what responsibilities face a community needing to support a legacy and a future of code lists?
- what tools are available to be adapted for use in validating code lists?

This book covers that subset of OASIS genericcode 1.0 suitable for expressing a single code list, and all of the OASIS context/value association draft specification

- <http://docs.oasis-open.org/codelist/genericcode>
- [http://www.oasis-open.org/committees/document.php?document\\_id=29990](http://www.oasis-open.org/committees/document.php?document_id=29990)

The first two chapters overview controlled vocabularies in general

- what do they represent?
- how are they used?
- how, historically, have they been deployed?
- what new approaches are available for deployment?

The third and fourth chapters overview the specification of code lists in XML

- the use of the OASIS genericcode specification
- the detail of the OASIS genericcode vocabulary

The fifth and sixth chapters overview the specification of context/value association files

- the use of OASIS CVA files
- the detail of the OASIS CVA vocabulary

The last chapter outlines how to adapt validation stylesheets to use genericcode and CVA files for an arbitrary vocabulary

The only annex introduces Crane Softwrights Ltd.'s OpenOffice 3 XML filter package

Crane-gc2ods

- opening and saving OASIS genericcode files
- opening and saving OASIS CVA files

Enjoy!

## Chapter 1 - Controlled vocabularies



- Introduction - XML document interchange
- Section 1 - Facets of controlled vocabularies

### Outcomes

- understand the role of enumerated values in business documents

## XML document interchange

Introduction - Chapter 1 - Controlled vocabularies



An XML document describes a hierarchy of information items

- XML is only responsible for representation of information and not the meaning of information
  - how information is labeled allows it to be identified, not interpreted
  - up to applications to interpret the meaning of the labels and information so-labeled
- each item is labeled using the document's XML vocabulary
  - the item's value is expressed in an attribute's specification or an element's text value
- document constraints describe limitations on the contents of the XML documents
  - what is allowed to be used as item labels and where
  - what is allowed to be used as item values and where
  - a document isn't XML unless it is well-formed
    - rules govern the proper labeling of the information in the hierarchy
    - labels can be comprised of namespace URI strings to be globally unique
      - the metaphor for "labels in a namespace" is "words in a dictionary"
  - different document constraint languages provide different validation features
    - directives of the language engage validation semantics

Business documents have many information items whose values are controlled

- code lists have been used for hundreds of years
  - show up in historical documents, business records, passenger manifests, etc.
- codes, identifiers, any information item with a predetermined value set
  - like a label, a code represents the semantic, it doesn't "mean" the semantic
  - nothing in the value conveys understanding, only representation
  - still up to an application recognizing the code to be pre-programmed to interpret the semantic associated with that code
  - sender and receiver need to agree on the understanding of the value
- the information's value is limited to one or more of a set of fixed values
- item values do not impact on the structure of the document

Two distinct kinds of "vocabularies" for interchange

- the XML vocabulary of element and attribute labels
- a controlled vocabulary of code or identifier values
- in document interchange, the vocabularies represent the concepts and information for commonly-understood semantics
- in applications, internal representations of both may be very much richer than the interchange representation

## Controlled vocabulary semantics

Introduction - Chapter 1 - Controlled vocabularies



A controlled vocabulary is the set of agreed-upon values for a concept

- e.g. the list of country code abbreviations
  - e.g. "CA" for Canada, "US" for United States
- e.g. the list of currency code abbreviations
  - e.g. "CAD" for Canadian dollars, "USD" for United States dollars
- e.g. the list of transaction payment means
  - e.g. "10" for cash, "20" for cheque
- e.g. the list of units of measure
  - e.g. "KGM" for kilogram, "MTQ" for cubic meter
- e.g. identifiers for different kinds of dimensions
  - e.g. representing gross weight and net weight
- e.g. a company's private list of product and service identifiers
  - e.g. catalogue part numbers

Each value in a controlled vocabulary represents a particular semantic

- for obvious enumerated concepts, no semantic need be published authoritatively
  - e.g. the directions of latitude are either "North" or "South" of the equator
  - e.g. currency conversion operators are either "Multiply" or "Divide"
- for public vocabularies, the associated semantic is a published concept
  - managed by a public authority recognized as the trusted custodian of values
    - e.g. the International Organization for Standardization (ISO) list of country codes, currency codes, container sizes, etc.
    - e.g. the United Nations Economic Commission for Europe (UN/ECE) list of port codes, types of payment means, etc.
    - e.g. the Canadian Post Office list of Canadian province and territory abbreviations
- between trading partners, the associated semantic is an agreed-upon concept
  - e.g. the list of identifiers representing product and service offerings of a vendor
  - e.g. the document status codes accepted by a particular work flow specification

All parties implicitly agree to interpret the concepts in the same way in their independent applications

- by constraining the expression of the possible values to an agreed-upon set, both parties set expectations for interchange
- a formal expression of the constraints can form part of a business contract agreeing to limit values used to only the agreed-upon set
- traditional approaches to using W3C schema conflate the document constraints with the value constraints
  - new approaches are needed to layer value constraints on document constraints
- an important caveat: "obvious" values may not be obvious to all

## Controlled vocabulary semantics (cont.)

Introduction - Chapter 1 - Controlled vocabularies



A controlled value is necessarily unique in a single given list

- if a given string value represented more than one concept it would be ambiguous and there would be no way to distinguish which concept was desired
- list meta data for a value distinguishes ambiguous values in combined lists
  - the values may overlap when meta data is used to identify which list's distinct value is being used
- the unique value is analogous to a relational database table key
  - used as a lookup value
- another use of the "words in a dictionary" metaphor
  - the meta data of the list defines which dictionary the words are from
  - the meta data may distinguish different versions of the same dictionary

Each controlled value may have many associated values

- value meta data may be simple strings or compound values
  - compound information can be expressed in rich markup
- analogous to relational database table columns
- display string(s)
- non-normative synonyms
- language translations
- supporting detail and nuance
- meta data
  - derivation method
  - source of information

ISO parlance has been in use a long time for code lists

- "Code" refers to a value's unique key value within its list
- "Name" refers to that value's description with which meaning is intended to be expressed
- for some concepts, far more information needs to be associated with values

## Controlled vocabulary semantics (cont.)

Introduction - Chapter 1 - Controlled vocabularies



Controlled vocabularies are used in documents, databases, applications, messages

- by controlling the representation of a concept, a specified value can unambiguously identify the associated semantic
  - provided all users of the value understand the concept in the same manner
  - the burden is on the trusted custodian of the values to maintain the documentation of the list
- abbreviated values (codes) may provide a savings of effort or space when otherwise the expression of the concept is long-winded or wordy
  - the abbreviation is consistent
  - mnemonic values are typically biased to a particular language
    - e.g. "USD" mnemonic for "United States Dollars"
    - e.g. "ES" mnemonic for "Spain" ("España" is Spanish for "Spain")
  - non-mnemonic numeric values are often used as representations of abstract concepts without language bias
    - e.g. "42" non-mnemonic for "Payment to bank account" payment means
  - the mnemonic or non-mnemonic abbreviation is typically short
    - e.g. "51" non-mnemonic for "norme 6 97-Telereglement CFONB (French Organisation for Banking Standards) - Option A" payment means
- commonly used for centuries in messages to keep messages succinct

Promotes consistent interpretation of the value

- all applications can follow the published or agreed-upon semantics
- opportunity for misinterpretation through neglect or accident
- if each trading partner came up with their own abbreviations independently, it would be impossible to know that two different values represent the same concept
- removes language dependencies when abbreviating the same concept in two languages
  - though some codes are mnemonically derived from a native language, the rule governing that prevents the code from being derived differently in another language
- meta data columns can include various translations
  - promotes common interpretation

## Codes and identifiers

Chapter 1 - Controlled vocabularies

Section 1 - Facets of controlled vocabularies



As a general rule of thumb (but not definitively), a controlled vocabulary information item is typically either a code or an identifier

- these are very symmetrically-defined constructs that are distinguished by arbitrary decisions of construction and use
  - guidelines and distinctions are not black and white
  - whether the values are characteristic or lookup can be twisted one way or the other
- these concepts are not always consistently applied
  - e.g. in UBL some identifiers could easily be codes and vice versa
- a code typically represents a unique concept, group or type using a *characteristic* value
  - e.g. a currency code for an account value - "GBP" (British pounds)
  - e.g. a unit of measure for a measurement - "MTR" (meters)
  - e.g. a shipping container's dimensions
    - this is an example of a set of coded values created by the application of a scheme on component parts of the value describing the container's height, width, depth and features
  - e.g. a method of transport
  - e.g. a document's type
- an identifier typically represents a unique thing or singleton from a group using a *lookup* value
  - may be synthesized by applying an algorithmic scheme
    - the range of identifier values may, however, be enumerated as members of a list
  - e.g. a particular account's identifier - "travel" or "supplies" or "ABC0001"
  - e.g. a particular dimension - "gross width" or "net width"
  - e.g. a particular aircraft's identifier
  - e.g. a particular catalogue item's identifier

Trading partners may wish to constrain either codes or identifiers or any other information item as a controlled vocabulary

- codes typically taken from a set representing known semantic concepts
- constraining an identifier would be from a fixed list of identifiers
  - e.g. a set of account identifiers
- open-ended identifiers would typically not be constrained
  - used to identify things that are being created



## Code list registration authorities

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



The custodians of abstract code lists are typically the authorities with governance

- users of a list have a level of assurance regarding the maintenance of the sets of values
- stability is implied by the authority's governance of the concepts and expressions of the list members

The publishing of the list is different from the definition of the list

- the authority selects and defines codes in the abstract based on semantics (meaning)
- the list of codes is published hopefully with sufficient information to convey the semantics they represent so that all users interpret the codes as meaning the same concepts

The authorities can publish their code lists in many possible formats

- prose lists and descriptions
  - text files
  - word processing files
  - web site pages (HTML files)
  - algorithmic descriptions (e.g. ISBN checksum)
  - value assemblies (e.g. container height, width, depth and feature values)
- W3C schemas with annotations
- Comma Separated Values (CSV) files
- database tables
- colloquial XML expressions
  - using a bespoke document model invented by a community of users
  - an XML vocabulary not standardized outside of the community or users
- standardized XML expressions
  - using a published document model created by a committee effort
  - openness of process and access to results are important in assessing protection against private interests or encumbrances

Alternative expressions of lists may be made available in the absence of bona fide expressions

- a stop-gap measure to make up for the authority not having published the information in a useful form
- e.g. UBL has expressed a number of published abstract code lists using XML syntax until such time as the official custodians publish their own artefacts for public use

## Identifying controlled vocabularies

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Some controlled vocabularies are already officially maintained

- custodians are typically international standards organizations
- e.g. currency codes by ISO (ISO 4217)
- e.g. country codes by ISO (ISO 3166-1)
- e.g. payment means codes by UN/ECE (UN/ECE 4461)

Projects must establish which codes are applicable to their work

- community responsibility
  - manage expectations of individuals and trading partners
  - guide community in common understanding of concepts and representations
- a subset of codes from established lists
  - don't re-invent the wheel
- new codes for use where an established list is deficient
  - are extensions needed for the community to use?
- new lists of codes where there are no established lists
  - are entire new lists required for a set of community semantics?

Where new codes are needed,

- what do each of them mean?
  - what meta data might be associated with each?
- how are they coded?
  - mnemonic? numeric? arbitrary?
- which values are unconstrained?

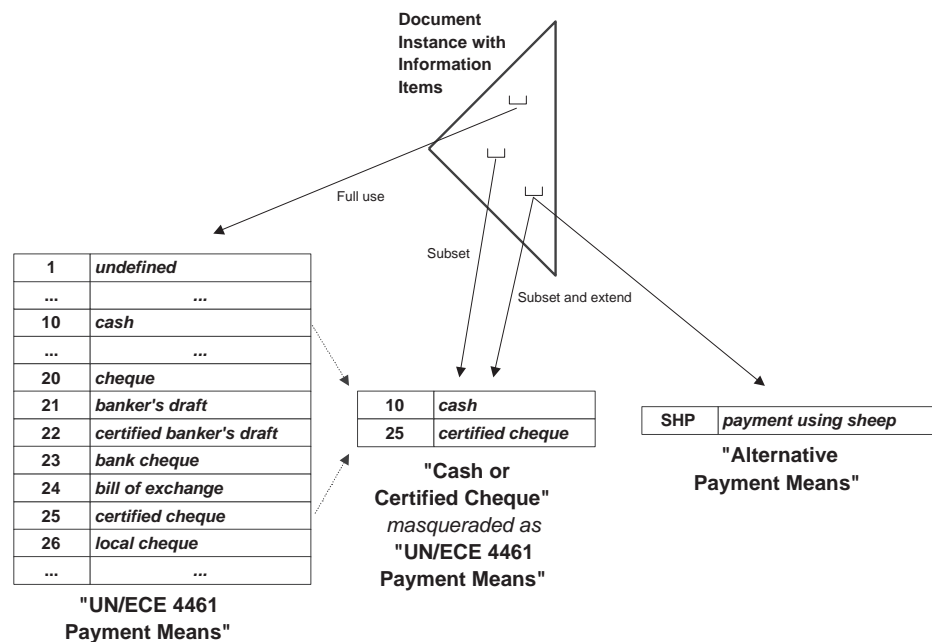
## Identifying controlled vocabularies (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



List-level meta data identifies the list being used

- the needed list is an established list
- the list identification is the official list-level meta data



## Identifying controlled vocabularies (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Value-level meta data qualifies the code with more detail

- code and name are not always sufficient to identify information
- value-level meta data can be used to distinguish facets of the code

Code	Name	Country	Subdivision	Port, rail, road, air, post, multi-modal, fixed, border	IATA	
ADALV	Andorra la Vella	Andorra		3,4,6	ALV	...
...	...	...	...	...	...	...
USCB8	Columbus	United States	MT	2,3	CB8	...
USCBW	Columbus	United States	WI	3	CBW	...
USCLU	Columbus	United States	IN	3,4	CLU	...
USCMH	Columbus	United States	OH	4	CMH	...
USCSG	Columbus	United States	GA	3,4	CSG	...
USCUS	Columbus	United States	NM	3,4,B	CUS	...
USCZX	Columbus	United States	NC	3,6	CZX	...
USOLP	Columbus	United States	MO	3,6	OLP	...
USOLU	Columbus	United States	NE	3,4	OLU	...
USUBS	Columbus	United States	MS	3,4	UBS	...
USUCU	Columbus	United States	KS	2,3	UCU	...
USVCB	Columbus	United States	TX	3	VCB	...
USVDA	Columbus	United States	MI	4	VDA	...
USYBC	Columbus	United States	NJ	2,3,6	YBC	...
...	...	...	...	...	...	...
USCB8	Columbus	United States		4	WKI	...

"UN/ECE Rec 16 LOCODE"

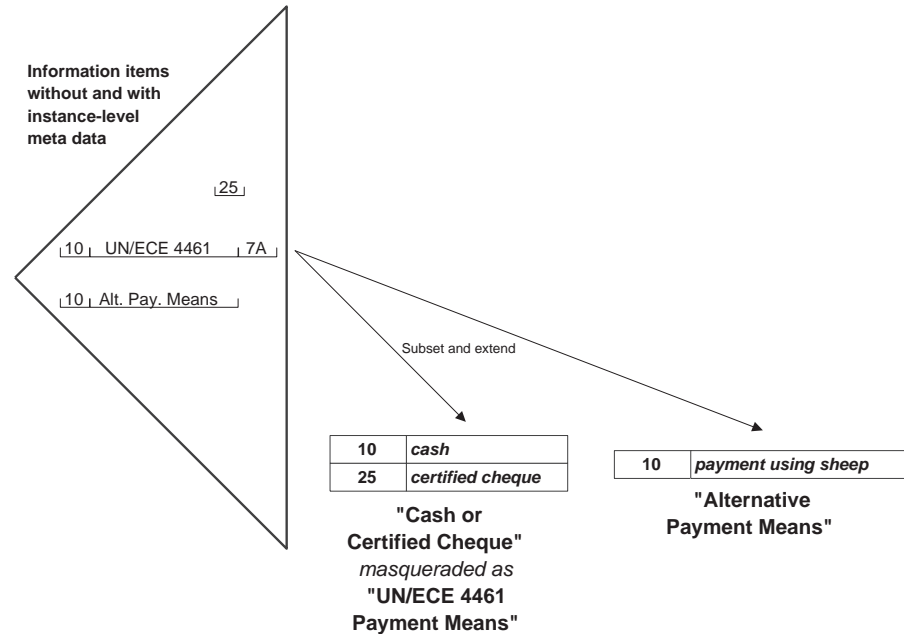
## Identifying controlled vocabularies (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Instance-level meta data qualifies the use of a code

- tells an application the list in which the code is found
- clarifies the meaning



Community responsibility when defining the XML vocabulary

- which instance-level meta data can the user specify?
- how does the user specify instance-level meta data?

## Identifying controlled vocabularies (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Summary of controlled vocabulary meta data

- list-level meta data
  - distinguishes one list of values from another list of values
  - responsibility of the controlled vocabulary custodian
- value-level meta data
  - distinguishes one value from another value within the same list
  - responsibility of the controlled vocabulary custodian
- instance-level meta data
  - distinguishes from which list a value is being used
  - responsibility of the XML vocabulary designer
  - utilized by the XML document writer

## Modeling controlled vocabularies

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



The organization of a set of associated values for all key values is tabular

- one row per semantic concept
- one or more key value columns uniquely identifying the concept
- zero or more meta data value columns defining the concept

The maintenance of list information necessarily needs to be tabular

- such a need distinguishes the available enumeration technologies as to their usefulness
  - i.e. a technology that does not support a tabular arrangement is not as useful as a technology that does support a tabular arrangement of list information

Maintaining an independent expression provides for re-use and change isolation

- the maintenance of a list of values will likely have a different life cycle than the contexts in which the values are used
- revising an external expression of a controlled vocabulary prevents having to change an expression in which a controlled vocabulary is embedded

Human language translations may help as supplemental information

- may reduce problems interpreting what a value represents

## Expressing controlled vocabularies

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Non-standard use of spreadsheets, word processing, comma-separated value (CSV) documents is common

- each custodial organization may have its own way of expressing the representation values and their associated semantics
- applications incorporating the values into their validation processes would need to accommodate ad hoc means with ad hoc measures
- the expression may not be well defined for maintenance

The interchange representation is independent of the internal representation

- though some applications may choose to use the interchange representation as the internal representation
- lookup strategies should be based on application requirements and be independent of interchange requirements

Artefacts for legal contractual agreements may be ambiguous

- using a standardized representation allows both parties to interpret all lists in the same fashion
- prose is often improperly used when meta data may be less ambiguous
  - especially when human language translation is involved

## Expressing controlled vocabularies (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Standards exist with which one enumerates the defined values of a controlled vocabulary

- the choice of expression empowers the use of that expression in different circumstances
- some XML designers fervently believe that document values belong in a document schema
- some XML designers fervently believe that document values must never be in a document schema

W3C Schema enumerations

- designed for use only in validation with W3C Schema semantics
- the formalism only captures the key coded values in a standardized structure
  - the associated meta data may be expressed in a non-standardized structure
  - only one key can ever be used to distinguish the information in the list
- document-wide scope of re-use
  - e.g. every use of the code list incorporates every code of the code list
  - using only a subset of codes requires declaring a separate code list for those contexts where the subset is needed
    - there is a question of what meta data to use for the lists
    - the full list's meta data would be inappropriate for a subset list, yet instances might require the use of the meta data for the full list
- the expressions are intertwined with the expressions of structural constraints
  - to change an enumeration one must "touch" the schema files that participate in structural validation
  - risk of inadvertently modifying the structural constraints, or burden of proving that the structural constraints were not inadvertently modified

OASIS Genericcode 1.0 (2007)

- designed for maintenance of the meta data of an enumeration and its members
- the formalism captures all information about values in a standardized structure
  - when there are multiple keys, the actual key needed can be chosen by an application
  - specifies standardized list-level meta data
    - all controlled vocabularies can be identified using the same mechanisms
  - specifies mechanisms for arbitrary value-level meta data
    - each controlled vocabulary can satisfy its own requirements for value distinction and definition
- context-free scope of use
  - the definition of the codes is independent of the specification of where the codes are used
- the external XML-based expression is independent of any particular use
  - useful for validation or user-interface definition or any use
- still a role for schema specification of available instance-level meta data

## Data entry of controlled vocabularies

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



By formally associating the use of value lists with document contexts, one can direct valid data entry

- directs a user interface
  - can be written to only allow entry of a value from the associated values
- value-level meta data is helpful
  - could be presented to the user to help them choose the right value to use in the data entry
- instance-level meta data records list-level meta data where needed for disambiguation
  - when one information item can have a value from two lists, and a code is needed that exists with the same value in each list, the list-level meta data distinguishes the code and needs to be recorded as instance-level meta data

## Application development supporting controlled vocabularies

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Controlled vocabularies are declarative while application program code is imperative

- easier (therefore cheaper?) to change an outboard declared list of allowed values than to change the inboard program logic
- non-programmer resources can be tasked with changing the declared vocabularies

An application can blindly support all values in a controlled vocabulary

- the application can support all possible allowed values and presume that pre-validation has rejected those instances where a supported value is not allowed
- the flexibility is in the filtering of allowed instances by dynamic application of value constraints during validation, without changing the programming in the application

The trading partner relationship constrains which values are allowed in a given transmission

- message filtering ensures only the messages with the allowed values for a given trading partner are passed for processing

Reduces application development to support new trading partners

- no need to change the program for every trading partner or new trading partners

Flexible to changing trading partner relationships

- as a relationship with a partner matures or changes, only the message filtering need change, not the application code

## Validating controlled vocabularies

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Having an expression of valid values enables the validation of specified values

- validation can reject an instance before engaging an application to act on the instance
- off-loads the validation responsibility (and possible error) from applications
- ensures consistent loading of database values

Values outside of the allowed set are considered invalid

- trading partners would not necessarily know what semantic an unexpected value represents
- legal agreements could not be entered into where the parties have arbitrary values possibly representing concepts outside of the agreement

Methodologies are published with which one confirms the proper selection of values in an XML information item

- traditional use of grammar- or type-based document schemas
  - e.g. XML DTD - grammar-based schema language
  - e.g. ISO/IEC 19757-2 RELAX-NG - grammar-based schema language
  - e.g. W3C Schema - type-based schema language
- alternative schema expressions
  - e.g. ISO/IEC 19757-3 Schematron - assertion-based schema language

Traditional approaches validate values at the same time as validating structure

- conflates structural validation with value validation
- inflexible to dynamically changing business requirements
  - no need to change the structural validation just because business relationships change
- business agreements impact on values but do not impact on document structures

## Validating controlled vocabularies (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies

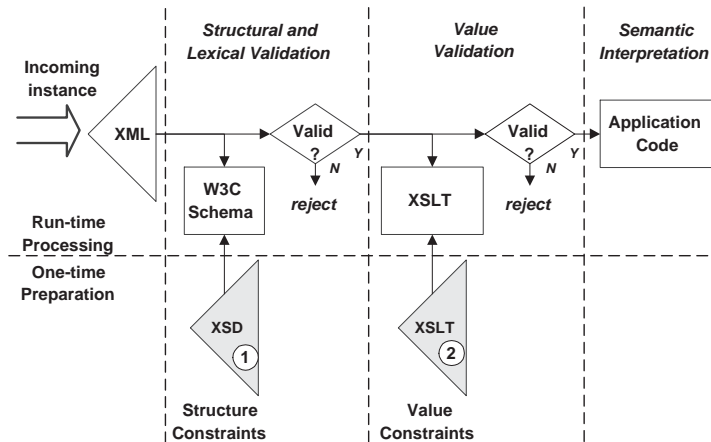


UBL 2.0 separates UBL conformance from code list conformance

- to which version of UBL schemas do the structures conform?
  - e.g. UBL-Invoice-2.0.xsd for an invoice
- to which code lists do the values conform?
  - e.g. defaultCodeList.xsl for a suite of typical code lists

Layering value constraints on top of structural and lexical constraints

- can be used for any XML document structure, not only UBL
- can be applied to any information item with an enumerated set of allowed values
- not restricted to only codes or identifiers
- can be built on top of ISO/IEC 19757-3 Schematron
- separates structural/lexical validation from value validation

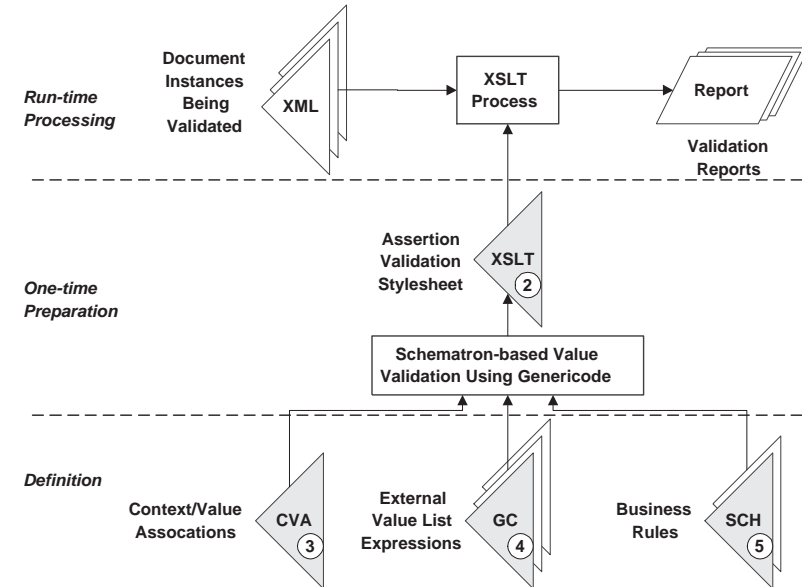


## Validating controlled vocabularies (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Opportunity to incorporate many kinds of value constraints



Context/value associations establish which code lists apply where in the document

- gives flexibility to specify different codes for the same conceptual value used in different document contexts

External value list expressions in genericcode

- the XML documents defining the controlled vocabularies
- includes list-level and value-level meta data

Business rules can express co-occurrence and algorithmic constraints

- more powerful than simple declarative approaches
- use Schematron for arbitrary XPath expressions

## Semantic representation by fixed values

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



### Assigned semantics

- each unique value represents an associated concept, label or longer value
- community of users agree on the association between specified value and represented value
- a value has two aspects of context in order to have some meaning
  - context of use/location
    - where in the document is the information item being used
  - context of definition/meaning
    - from which set of values is the information item value being obtained
    - without explicit context a value may be ambiguous or reliant on informal agreement
- especially important for non-mnemonic codes: e.g. "42" vs. "USD"
  - the meaning of some mnemonic codes might be guessed based on context of use, e.g. "USD" for a currency
  - non-mnemonic codes typically have no basis for guessing the meaning, e.g. "42" for a payment means

### Instance-level meta data disambiguates a code when context is insufficient

- used for identification of values and definition of values
- list meta data identifies the collection from which the value is taken
  - information about the collection as a whole
  - gives context to the specified values
- value meta data helps define the semantics or details of the value itself
  - information about the one particular coded value

### Changes in time can affect the interpretation/semantics of values

- the collection of values evolves creating a new version of an existing code list
  - identified by associated meta data
- the meaning of individual values evolves
  - described by associated meta data or prose
- migrating data from old to new may require simultaneous support of multiple versions of the same code list
  - requires flexibility not typically associated with traditional schema-based approaches to using codes
- unused and retired codes might get re-used later for new semantic concepts
  - e.g. country code "CS" was Czechoslovakia before 2003 and was reserved in 2006 for Serbia and Montenegro

## Trading partners and agreements

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



### Trading partners need to agree on the structure and content of interchanged documents

- so doing provides interoperability between independent systems acting on the information
- using XML isn't a magic bullet
  - it doesn't make our programs better, it makes our interchange of information more reliable
  - layers interchange constraints on top of implementation foundations
- structuring the information in a standardized fashion ensures the information is communicated
  - where to find information based on how it is labeled and where it is found hierarchically
- agreeing on the semantics behind values in the communicated information ensures the information is understood
  - what specified information values mean and represent in the abstract

### Relationships between trading partners change, while standards do not

- trading partner requirements can be layered on top of industry standards
- trading partners can also anticipate future changes to standards

### Published interchange specifications cannot pretend to know every value that trading partners need

- there is no standardization of many business concepts, just practical and pragmatic use for those concepts of import to trading partners
- therefore that can be no standardization of a set of values representing business concepts that are particular to trading partners

### Industries can state what the semantics are behind values

- trading partners can agree on which values to use



## Trading partners and agreements (cont.)

Chapter 1 - Controlled vocabularies  
Section 1 - Facets of controlled vocabularies



Codes for some established business practices can be supplemented or subset by trading partners

- e.g. extending document status codes
  - a typical workflow will have typical status values for the progress of a document
  - particular workflow systems used by trading partners may use only some or maybe more document status values for a given document
- e.g. restricting payment means codes
  - payment can only be by certified cheque or credit card
- e.g. restricting and extending transportation status codes
  - the suite of status codes includes a subset of a standard suite in combination with non-standard additional values

Identifiers are especially important as they specify actual business objects and not business concepts

- e.g. account identifiers
  - every business will probably have a different set of accounts than other businesses
  - when engaging in a transaction, a trading partner can publish its accepted list of account identifiers so that a correspondent knows which values can be used
- e.g. measurement identifiers
  - a catalogue item's characteristics need to be identified unambiguously (e.g. "gross weight" is distinct from "net weight")

Additional business rules can be layered on top of value constraints

- e.g. validity of non-coded data values based on trading partner relationship
  - e.g. a maximum value for an order
- e.g. the nature of a product identified by its identifier may restrict the payment means by which it is paid for
  - e.g. no credit cards for certain products

## Chapter 2 - Defining and using controlled vocabularies



- Introduction - Controlled value list maintenance and identity
- Section 1 - XML value specification and validation
- Section 2 - Example: controlled vocabularies in UBL
- Section 3 - Declarations of controlled vocabularies

### Outcomes

- review enumerated values in UBL information items
- consider examples of meta data when specifying enumerated values from controlled vocabularies

## Controlled value list maintenance and identity

Introduction - Chapter 2 - Defining and using controlled vocabularies



A list of values has an identity in the abstract, regardless of how it is maintained

- e.g. ISO 3166-1 country codes
- e.g. UN/ECE 4461 payment means codes
- e.g. UBL 2.0 document status codes

The complete list may be maintained by hand or by a database or by any means

- the management of the values is important to long-term maintenance
- some lists may have tens of thousands of entries (e.g. vehicle model codes)
- a list may be synthesized by an algorithmic process
  - e.g. the 100 ISBN numbers assigned to publisher "978-1-894049"

The concrete expressions of the lists may vary based on purpose or contextual use

- e.g. complete lists
- e.g. restricted subset lists
- each list and list subset expression must necessarily be uniquely identified
  - a subset of a list cannot have the same identity as the complete list otherwise there would be confusion regarding which list is the "true" list
  - identity can be expressed as meta data for the list or list subset
- the concrete expression may take any useful form for the user
  - e.g. simple text
  - e.g. comma-separated values
  - e.g. XML files
  - having a standardized representation of lists would encourage the development of more widely-useful applications

The sender and receiver may have different identities for a list of identical values

- e.g. the sender specifies an ISO country code
  - the meta data for the list is that of the complete list
- e.g. the receiver only accepts a subset of ISO country codes as valid
  - the meta data for the subset list is necessarily different than that of the complete list
  - for validation purposes the subset list must masquerade as the complete list yet reject specified values outside of the subset list

## Controlled value specification

Introduction - Chapter 2 - Defining and using controlled vocabularies



A controlled value is, in fact, a multi-faceted value

- the list from which the code value is obtained
  - described by meta data for the list
  - the list identification itself may be multi-faceted
- the key code value itself
  - unique within any given list
- properties (value-level meta data) of the values themselves
  - helpful in understanding the semantics of the key code value

When an information item can be populated with a coded value, it should also be possible to specify the associated value list meta data

- even very stable lists of values will change over time
  - one may need to specify a chronologically-distinct interpretation of a given value
- e.g. the list of provinces in Canada changed in 1999 when the Northwest Territories was split into two territories: Nunavut and the Northwest Territories
  - the Canadian postal province and territories indication of the Northwest Territories was and remains "NT" even though the definition of the territory changed
  - if the distinction is important to a trading partner, then provision for making the distinction must be made available
- e.g. the list of country codes changes frequently
  - before 2003 "CS" represented Czechoslovakia and since 2006 "CS" is reserved for Serbia and Montenegro
- one information item value may be an item selected from one of a number of lists
  - if all of the values are mutually exclusive in separate lists, there is no risk of confusion other than changes over time for any given list as noted above
  - if the values in the lists are not mutually exclusive, meta data is required to disambiguate an ambiguous specified value

The risk is borne by the party encoding the information that the recipient can properly decode the intent expressed by the information

- list meta data is often optional and is often ignored when coded values are specified
- the more specific a specification is, the less opportunity for improper understanding of the intended meaning

## Controlled XML information item value specification

Chapter 2 - Defining and using controlled vocabularies  
Section 1 - XML value specification and validation



Lists evolve over time and are published in many versions

- different versions of lists have different available values
- a different version of a list may redefine an old value as having a new meaning

One needs to be able to specify both the coded value and any associated list-level meta data

- necessitates a compound structure in the XML markup for instance-level meta data
- the recipient can then disambiguate a value determined to be ambiguous
  - ambiguous because the value exists in two different lists thus with two different semantic associations
- without the information there is a risk the recipient misinterprets the sender's intent

E.g. UN/CEFACT Core Component Technical specification (CCTS)

- unqualified data types provide for varying amounts of instance-level meta data for different code and identifier information items
- e.g. UBL uses CCTS unqualified data types and inherits its code and identifier meta data
- a given coded information item may be an element or may be an attribute
- decision to use attached attributes to capture the associated instance-level meta data
  - UN/CEFACT determined that not all lists need wide-ranging flexibility in associated code list identification meta data

There may need to be a strategy to omit meta data from a value

- an application may anticipate the future standardization of a value
  - in the short term the value can be defined in an ad hoc list used in conjunction with the old standardized list
  - avoiding meta data avoids asserting the new value is in any standardized list, as the standardized list meta data might trigger a constraint violation
  - when the new publication of the revised list has the new value, the ad hoc list can be abandoned
- runs the risk that future standardization does, in fact, use the anticipated value for the anticipated semantic

## Document context

Chapter 2 - Defining and using controlled vocabularies  
Section 1 - XML value specification and validation



An important aspect of XML structure is the hierarchical document context for an item

- a labeled information item at one place in the document tree has a different meaning than an identically-labeled information item found elsewhere in the document tree

Document context can be widely specified (by scope) or narrowly specified (by address)

- consider the following example where all uses of `currencyID=` happen allow the same set of currency values (though in practice this may not indeed be a business case)
- consider that the controlled values for `<cbc:ID>` information item are different when the item is a child of `<cac:TaxCategory>` than when a child of `<cac:TaxScheme>`
  - the label is the same but the meaning is different

```

01 <cac:TaxSubTotal>
02   <cbc:TaxableAmount currencyID="USD">100.00</cbc:TaxableAmount>
03   <cbc:TaxAmount currencyID="USD">15.00</cbc:TaxAmount>
04   <cac:TaxCategory>
05     <cbc:ID>Z</cbc:ID>
06     <cbc:Percent>15</cbc:Percent>
07     <cac:TaxScheme>
08       <cbc:ID>Provincial Tax</cbc:ID>
09     </cac:TaxScheme>
10   </cac:TaxCategory>
11 </cac:TaxSubTotal>

```

Partial-document context specification is needed for documents interchanged between trading partners

- the business rules for the same information item in different document contexts may be different thus requiring different value validation constraints
  - structural constraints are unchanged and are reliably validated against the document structure

W3C schema enumerated data types assigned to globally-defined constructs only have document-wide context

- this makes W3C schema enumerations unsuitable for some business document validation

## Controlled value information item validation

Chapter 2 - Defining and using controlled vocabularies  
Section 1 - XML value specification and validation



### Structural, lexical and value validation distinction in regard of business/value rules

- constraining an information item to its place in the structural hierarchy and in its lexical structure is typically independent of business/value rules
  - suitable to put such constraints in a published and standardized read-only schema
  - esoteric structure requirements can be expressed if necessary as a business rule
    - e.g. "for addresses in the US, the building name cannot be specified"
- constraining the value of an information item can be very dependent on trading partner business/value rules
  - e.g. list of account identifiers
  - e.g. list of acceptable currency codes
  - unsuitable to put such constraints in a published and standardized read-only schema
  - value constraints should be layered on top of structural and lexical constraints so they can be modified without modifying the structural constraints
  - the structural and lexical constraints are still obligatory as the value constraints mean nothing if the structural and lexical constraints are violated

## Controlled value information item validation (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 1 - XML value specification and validation



### Direct program validation

- traditional programming practice leaves structural, lexical and value constraint checking to individual programs
  - modularization of code provided compatibility between applications
  - applications based on different code needed different implementations
  - applications written by different parties employed different implementations
  - opportunities abound for inconsistencies in the implementations of constraint checking
    - errors in implementation
    - errors in interpretation of requirements
- programs need to reflect (possibly changing) trading partner relationships
  - maintenance required for business reasons as well as technical reasons

### Outboard validation

- traditional XML document validation implements structural and lexical constraint checking using schema expressions
  - an outboard schema expression describes the constraints
  - publicly-available code fragments implement constraint checking
- traditional XML document validation implements value checking using schema expressions
  - enforces document-wide context on globally-defined constructs which may be inappropriate for trading partner business rules
- necessary to abandon traditional value checking and utilize a layered approach
  - changing the value constraints does not change the structural and lexical constraints
  - trading partners can tailor the value constraints to business requirements
- simplified program development can support all codes
  - the outboard validation serves as a filter for messages changing their nature
    - program maintenance reduced to only technical reasons
    - business reasons expressed in value validation

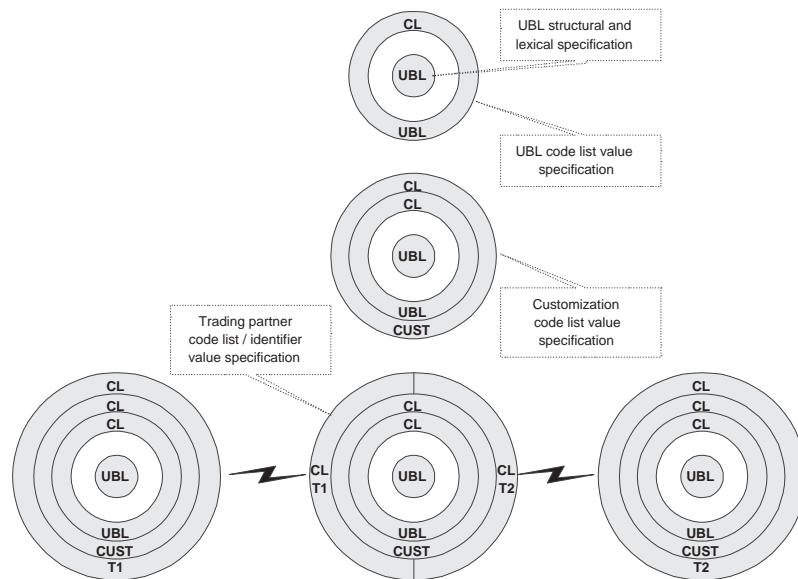
## Layered constraints on business documents

Chapter 2 - Defining and using controlled vocabularies  
Section 1 - XML value specification and validation



Many parties contribute to the value validation requirements of business documents

- illustrated by the requirements of UBL, a UBL customization and trading-partner-specific value constraints
  - UBL publishes a set of W3C schemas for structural and lexical constraints on values
  - UBL publishes a set of value constraints on particular information items
    - e.g. document status codes
  - a user community (e.g. the country of Denmark, the North European Subset, etc.) publishes a set of value constraints on particular information items
    - e.g. payment means codes
  - trading partners publish a set of value constraints on particular information items
    - e.g. account identifiers



## Example: controlled vocabularies in UBL

Chapter 2 - Defining and using controlled vocabularies  
Section 2 - Example: controlled vocabularies in UBL



The Universal Business Language (UBL) model is an example to be followed by other vocabularies

- characterized by the need for trading-partner flexibility and community definition

Structural and lexical constraint validation is standardized and supplied

- published normative W3C Schema expressions

UBL provides a non-normative starter kit of sets of code lists for only a few coded values

- trading partners should maintain compatibility with the starter kit of code lists
  - when possible only use extension and restriction of starter codes rather than replacement
  - a closed community of users may choose to use only their exclusive sets of values
- trading partner code lists can use or build on the UBL code lists
- no starter sets for identifiers are provided
  - up to trading partners to decide what identifiers are meaningful in their exchanges
- many lists are defined by UBL as empty
  - trading partners can use the empty shells to create their own agreed-upon set of values
- a small number of predefined lists are supplied
  - easily-agreed-upon immutable concepts such as arithmetic operators and compass directions
  - starter sets of basic concepts such as document status
  - compatibility sets of imported enumerations
    - UBL has adopted the Core Components Technical Specification XSD schemas
    - schemas are supplied with built-in XSD enumerations

Starter value validation is supplied for the starter kit of sets of codes

- the default validation supplied as part of the UBL package is only for code list coded values for those code lists for which the UBL TC provides values
- no use of document context, thus mimicking document-wide XSD schema validation

UBL also supplies code lists of values that are not engaged by default

- available in case they are of use to certain trading partners
- e.g. shipping container sizes/types and location codes

## Example: controlled vocabularies in UBL (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 2 - Example: controlled vocabularies in UBL



Predefined lists of codes are provided for immutable concepts

- e.g. arithmetic operators (multiply and divide)
- e.g. compass directions (north, south, east and west)

Predefined lists of codes are provided for basic concepts

- e.g. document status (cancelled, disputed, revised, etc.)
- e.g. line status (cancelled, disputed, revised, etc.)
- e.g. chip code (chip vs. magnetic stripe)

Predefined lists of codes are provided for published reference lists

- e.g. payment means (cash, cheque, credit card, etc.)
- e.g. channel codes (internal mail, postbox number, electronic mail, etc.)
- e.g. packaging codes (box, drum, bag, bunch, etc.)
- e.g. transport mode (maritime, rail, road, multimodal, etc.)

Non-extensible lists of codes are provided for CCTS supplementary components

- CCTS assumes the following values are checked using schema-expressed enumerations
  - currencies for amounts (CAD, USD, GBP, DKK, etc.)
  - units of measure for quantities and measures (meter, square meter, cubic meter, etc.)
  - MIME types for binary objects (image/jpeg, image/naplps, application/pdf, etc.)
- trading partners can constrain these sets of values to smaller sets of values
  - these sets cannot be augmented to include values not in the hardwired schema-expressed lists
  - any augmentation would trigger schema validation errors
- anticipating in UBL 2.1 that CCTS code lists will be extensible
  - removing W3C schema code list enumerations

Available code lists for interested parties

- shipping container types and sizes
- location codes (rail, road, IATA, etc.)

All other code lists and identifiers are declared in the schema with only the lexical constraint of being a normalized string, and without declaring a bounded set of values

- a "normalized string" is a string where consecutive sequences of white-space characters are replaced with a single space character
  - white-space characters in parsed XML are space, tab, carriage-return and linefeed
- trading partners can agree on any set of coded values to use for a given information item
  - provided the values are lexically acceptable for a normalized string

## Example: controlled vocabularies in UBL (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 2 - Example: controlled vocabularies in UBL



defaultCodeList.xsl provides code list conformance validation supporting the following:

- with the corresponding genericcode file name in the UBL 2.0 delivery
- UN/ECE Recommendation 19 Transport Mode Code
  - cl/gc/default/TransportModeCode-2.0.gc
  - incorrectly documented in UBL 2.0 files as recommendation 16
- UN/ECE Recommendation 20 Unit of Measure Codes
  - cl/gc/cefact/UnitOfMeasureCode-2.0.gc
- UN/ECE Recommendation 21 Packaging Type Code
  - cl/gc/default/PackagingTypeCode-2.0.gc
- UN/ECE Recommendation 24 Transportation Status Codes
  - cl/gc/default/TransportationStatusCode-2.0.gc
- UN/ECE 3155 Communication Address Code Qualifier
  - cl/gc/default/ChannelCode-2.0.gc
- UN/ECE 4461 Payment Means
  - cl/gc/default/PaymentMeansCode-2.0.gc
- UN/ECE 4465 Adjustment Reason Description
  - cl/gc/default/AllowanceChargeReasonCode-2.0.gc
- UN/ECE 8053 Equipment Type Code Qualifier
  - cl/gc/default/TransportEquipmentTypeCode-2.0.gc
- IANA\_7\_04 Binary Object MIME Code
  - cl/gc/cefact/BinaryObjectMimeCode-2.0.gc
- ISO 3166-1 Country Codes
  - cl/gc/default/CountryIdentificationCode-2.0.gc
- ISO 4217 Alpha Currency Codes
  - cl/gc/cefact/CurrencyCode-2.0.gc
- UBL chip codes
  - cl/gc/default/ChipCode-2.0.gc
- UBL document status codes
  - cl/gc/default/DocumentStatusCode-2.0.gc
- UBL latitude direction codes
  - cl/gc/default/LatitudeDirectionCode-2.0.gc
- UBL line status codes
  - cl/gc/default/LineStatusCode-2.0.gc
- UBL longitude direction codes
  - cl/gc/default/LongitudeDirectionCode-2.0.gc
- UBL operator codes
  - cl/gc/default/OperatorCode-2.0.gc
- UBL substitution codes
  - cl/gc/default/SubstitutionStatusCode-2.0.gc



## Declaration techniques

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



UBL 2.0 illustrates different ways of declaring coded value constraints

- described in the UBL Naming and Design Rules (NDR)
  - NDR rules govern the synthesis of W3C Schema XSD expressions of model constraints
- 91 code lists cover 98 code list-based information items in the library
  - 18 code lists have defined constrained values
  - 73 code lists are meta-data only and unconstrained
- W3C schema-declared enumerations inherited from UN/CEFACT CCTS approach to base types
  - suitable only for restricting by externally-specified values
- declaration of unconstrained code list-based information item data type
  - suitable for extending and restricting by externally-specified values
- declaration of unconstrained identifier-based information item data type
  - suitable for extending and restricting by externally-specified values
- layered application of genericcode-declared enumerations of code lists for values
  - a suitable document format for externally-specified values

UBL 2.1 will be taking the UN/CEFACT lists out of the W3C schema XSD files

- already some of the lists are out of date because they are explicitly codified
- using a single approach for all code lists will help keep deployments more consistent
- abandoning the schema-based approach will help keep deployments more flexible

## UN/CEFACT controlled-vocabulary declarations

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



W3C schema-declared enumerations of code list-based information item data types

The UBL TC committee agreed to utilize UN/CEFACT CCTS unqualified data types "off-the-shelf"

- UN/CEFACT ATG definitions of types are published using W3C schema enumerations
  - e.g. the unqualified data type `AmountType` has a `currencyID=` attribute
    - constrained by the `CurrencyCodeContentType` enumerated data type
- only three W3C schema enumerations are used
  - currency codes of amounts, MIME type codes of binary objects, units of measure for quantities and measures
- one W3C schema enumeration is provided in the UN/CEFACT fragments but is not used
  - language code
- see page 48 for an overview of UN/CEFACT code list meta data

Excerpt from

[http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/common/CodeList\\_CurrencyCode\\_ISO\\_7\\_04.xsd](http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/common/CodeList_CurrencyCode_ISO_7_04.xsd):

```

01 <xsd:simpleType name="CurrencyCodeContentType">
02   <xsd:restriction base="xsd:token">
03     <xsd:enumeration value="AED">
04       <xsd:annotation>
05         <xsd:documentation>
06           <ccts:CodeName>Dirham</ccts:CodeName>
07           <ccts:CodeDescription></ccts:CodeDescription>
08         </xsd:documentation>
09       </xsd:annotation>
10     </xsd:enumeration>
11     <xsd:enumeration value="AFN">
12       <xsd:annotation>
13         <xsd:documentation>
14           <ccts:CodeName>Afghani</ccts:CodeName>
15           <ccts:CodeDescription></ccts:CodeDescription>
16         </xsd:documentation>
17       </xsd:annotation>
18     </xsd:enumeration>
19     <xsd:enumeration value="ALL">
20       <xsd:annotation>
21         <xsd:documentation>
22           <ccts:CodeName>Lek</ccts:CodeName>
23           <ccts:CodeDescription></ccts:CodeDescription>
24         </xsd:documentation>
25       </xsd:annotation>
26     </xsd:enumeration>
27     <xsd:enumeration value="AMD">
28     ...

```

## UN/CEFACT instance-level meta data

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



## CCTS 2.01 declaration of unconstrained code list-based information item data type

All other code lists are declared only with lexical constraints, not with value constraints

- the lexical structure of the text value is constrained to be a normalized string
  - no leading spaces, no trailing spaces, all sequences of white-space characters are only a single space each
- meta data attributes identify the code list in which the value is found and defined

Excerpt from

<http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/common/UnqualifiedDataTypeSchemaModule-2.0.xsd>

- instance-level meta data is defined by UN/CEFACT and utilized unchanged by UBL

```

01 <xsd:complexType name="CodeType">
02   <xsd:annotation>
03     <xsd:documentation xml:lang="en">
04       ...
05     </xsd:documentation>
06   </xsd:annotation>
07   <xsd:simpleContent>
08     <xsd:extension base="xsd:normalizedString">
09       <xsd:attribute name="listID" type="xsd:normalizedString"
10         use="optional"/>
11       <xsd:attribute name="listAgencyID"
12         type="xsd:normalizedString" use="optional"/>
13       <xsd:attribute name="listAgencyName" type="xsd:string"
14         use="optional"/>
15       <xsd:attribute name="listName" type="xsd:string"
16         use="optional"/>
17       <xsd:attribute name="listVersionID"
18         type="xsd:normalizedString" use="optional"/>
19       <xsd:attribute name="name" type="xsd:string"
20         use="optional"/>
21       <xsd:attribute name="languageID" type="xsd:language"
22         use="optional"/>
23       <xsd:attribute name="listURI" type="xsd:anyURI"
24         use="optional"/>
25       <xsd:attribute name="listSchemeURI" type="xsd:anyURI"
26         use="optional"/>
27     </xsd:extension>
28   </xsd:simpleContent>
29 </xsd:complexType>

```

## UN/CEFACT instance-level meta data (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



## Declaration of unconstrained identifier-based information item data type

Identifiers are constrained very similarly to coded values

- lexically constrained to be a normalized string
- meta data attributes identify the scheme by which the identifier is constructed or enumerated

Excerpt from

<http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/common/UnqualifiedDataTypeSchemaModule-2.0.xsd>

- instance-level meta data is defined by UN/CEFACT and utilized unchanged by UBL

```

01 <xsd:complexType name="IdentifierType">
02   <xsd:annotation>
03     <xsd:documentation xml:lang="en">
04       ...
05     </xsd:documentation>
06   </xsd:annotation>
07   <xsd:simpleContent>
08     <xsd:extension base="xsd:normalizedString">
09       <xsd:attribute name="schemeID" type="xsd:normalizedString"
10         use="optional"/>
11       <xsd:attribute name="schemeName" type="xsd:string"
12         use="optional"/>
13       <xsd:attribute name="schemeAgencyID"
14         type="xsd:normalizedString" use="optional"/>
15       <xsd:attribute name="schemeAgencyName" type="xsd:string"
16         use="optional"/>
17       <xsd:attribute name="schemeVersionID"
18         type="xsd:normalizedString" use="optional"/>
19       <xsd:attribute name="schemeDataURI" type="xsd:anyURI"
20         use="optional"/>
21       <xsd:attribute name="schemeURI" type="xsd:anyURI"
22         use="optional"/>
23     </xsd:extension>
24   </xsd:simpleContent>
25 </xsd:complexType>

```



## Genericcode controlled-vocabulary declarations

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



### Genericcode-declared enumerations of code lists

- synthesized and managed outside of any structural and lexical constraint expression

#### Excerpt from

<http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/default/PaymentMeansCode-2.0.gc>:

```
01 <gc:CodeList xmlns:gc="http://docs.oasis-open.org/
02 codelist/ns/genericcode/1.0/">
03   <Identification>
04     <ShortName>PaymentMeansCode</ShortName>
05     <LongName xml:lang="en">Payment Means</LongName>
06     <LongName Identifier="listID">UN/ECE 4461</LongName>
07     <Version>D03A</Version>
08     <CanonicalUri>urn:oasis:names:specification:ubl:codelist:gc:
09 PaymentMeansCode</CanonicalUri>
10     <CanonicalVersionUri>urn:oasis:names:specification:ubl:codelist:
11 gc:PaymentMeansCode-2.0</CanonicalVersionUri>
12     <LocationUri>http://docs.oasis-open.org/ubl/os-ubl-2.0/cl/gc/
13 default/PaymentMeansCode-2.0.gc</LocationUri>
14     <Agency>
15       <LongName xml:lang="en">United Nations Economic Commission for
16 Europe</LongName>
17       <Identifier>6</Identifier>
18     </Agency>
19   </Identification>
20   ...
21   <SimpleCodeList>
22     ...
23     <Row>
24       <Value ColumnRef="code">
25         <SimpleValue>2</SimpleValue>
26       </Value>
27       <Value ColumnRef="name">
28         <SimpleValue>Automated clearing house credit</SimpleValue>
29       </Value>
30     </Row>
31     <Row>
32       <Value ColumnRef="code">
33         <SimpleValue>3</SimpleValue>
34       </Value>
35       <Value ColumnRef="name">
36         <SimpleValue>Automated clearing house debit</SimpleValue>
37       </Value>
38     </Row>
39     ...
40   </SimpleCodeList>
41 </gc:CodeList>
```

## Examples and locations of UBL code list definitions

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



### Many UBL information items are described by code lists

- <http://docs.oasis-open.org/ubl/os-UBL-2.0-update/cl/gc/>
- support for a prescribed collection of lists is provided by UBL TC
  - validated using
    - <http://docs.oasis-open.org/ubl/os-UBL-2.0/val/defaultCodeList.xsl>
- a constrained set of CCTS-specified values available only to be restricted
  - <http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/cefact/> directory
  - e.g. Currency\_ Code. Type
    - initially defined set of values to be all currencies of the world
- a predefined set of values available to be supplemented or restricted
  - <http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/default/> directory
  - e.g. Document Status\_ Code. Type
    - values "Cancelled", "Disputed", "NoStatus", and "Revised"
  - e.g. Payment Means\_ Code. Type
    - values "10" (cash), "20" (cheque), etc.
- a limited set of values unlikely to be changed
  - <http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/default/> directory
  - e.g. Latitude Direction\_ Code. Type
    - values "North" and "South"
  - e.g. Longitude Direction\_ Code. Type
    - values "East" and "West"
- no predefined values but available to be defined between trading partners
  - <http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/default/> directory
  - e.g. Code. Type for Invoice. Invoice\_ Type. Code
  - by far most of the code lists are defined only with meta data in this fashion
  - from a theoretical perspective, these lists are the lists of all possible codes
    - when two trading partners agree on a set of codes for these lists, they are in fact restricting the infinite set of codes to a finite set of codes
- predefined values available to trading partners but not engaged by default
  - <http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/special-purpose/> directory
  - not all users of UBL need these codes, but if they are available if needed
  - e.g. Container Size Type\_ Code. Type

## Instance-level meta data in XML instances

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



One can qualify the value for an information item defined by a code list or identifier

- the qualification is accomplished using meta data attributes on the element with the value
- all meta data is optional

Easiest (but riskiest) not to use instance-level meta data with codes

- most uses of codes do not need to be disambiguated
- recommended to use instance-level meta data if values from different lists are ambiguous

One can specify only as much instance-level meta data as desired

- it does not have to be "all or nothing"
- adding meta data to the value makes the specification of the value more distinguished for the sender
  - the sender is intending to distinguish the value as being not just any value but a particular value from a list so identified
- adding meta data to the value makes the validation of the value more stringent for the recipient
  - the recipient can ensure that the value received isn't just any value but is a particular value from an expected code list
- validation of items without meta data is purposely lax so as not to require meta data be specified
  - validation of meta data only happens in the presence of meta data

Specifying no meta data can be a strategy

- in anticipation of an as-yet-added entry to a standardized list
- the future extension can be specified in an ad hoc list that extends the standard list
- by not specifying meta data, validation does not attempt to constrain from which list the value is obtained
- once the standard list incorporates the new value, the ad hoc list can be abandoned

## Instance-level meta data in XML instances (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



Consider the difference between the two following values in a UBL instance:

- country code example: CS
  - prior to 2003 this represented Czechoslovakia
    - codes used now are CZ for Czech Republic and SK for Slovakia
    - simple retiring of one code and birth of two new codes
  - as of 2006 the country code "CS" has been reserved for Serbia and Montenegro
    - not so simple change in the semantics with the re-use of the same code
- country sub-entity code example of Canadian territories:
  - not so simple change in the semantics with the re-use of the same code
  - `<CountrySubentityCode listVersionID="1">NT</CountrySubentityCode>`
  - `<CountrySubentityCode listVersionID="2">NT</CountrySubentityCode>`
- not specifying a version would make the value ambiguous
  - `<CountrySubentityCode>NT</CountrySubentityCode>`

Instance-level meta data is needed to specify the appropriate associated list-level meta data

- specifying the version of the list ensures a sending application that a distinction is being made with a clear specification of precisely which semantics associated with the given value are intended
- detecting the version of the list would ensure that a receiving application is aware of the distinction being specified to know precisely which semantics are intended

## Instance-level meta data in XML instances (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



The instance-level meta data varies slightly for UN/CEFACT-defined unqualified data types

- inherited by those UBL information items derived from CCTS unqualified types
- the core component value is the value of the element
- the supplementary component value is an attribute of the element
- the meta data values are in other attributes of the same element
- for currencyID= of amounts
  - currencyCodeListVersionID=
    - mapped in CVA2sch to genericcode <Version>
- for unitCode= of the <MeasureType> element
  - unitCodeListVersionID=
    - mapped in CVA2sch to genericcode <Version>
- for unitCode= of the <QuantityType> element
  - unitCodeListID=
    - mapped in CVA2sch to genericcode <Version>
  - unitCodeListAgencyID=
    - mapped in CVA2sch to genericcode <Agency><Identifier>
  - unitCodeListAgencyName=
    - mapped in CVA2sch to genericcode <Agency><LongName>

Consider how currency values are entered in a UBL instance

- e.g. <cbc:Amount currencyID="RON">10.00</cbc:Amount>
  - specifies an amount of 10 Romanian new leu
  - no instance level meta data is included in the element specification
  - the recipient must make an assumption about which code list the value comes from
- e.g. <cbc:Amount currencyID="RON" currencyCodeListVersionID="1951">10.00</cbc:Amount>
  - first Romanian leu in 1867
  - Romanian new leu "RON" in 1947
  - in 1952 "RON" was replaced with "ROL" for Romanian leu
  - in 2005 "ROL" was replaced with "RON" for Romanian new leu
  - 1 RON(2005) is worth over 200,000 RON(1951)

## Instance-level meta data in XML instances (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



UBL-defined code list element meta data attributes summarized from page 42

- listID=
  - mapped in CVA2sch to genericcode <LongName @Identifier='listID'> or just the first <LongName>
- listAgencyID=
  - mapped in CVA2sch to genericcode <Agency><Identifier>
- listAgencyName=
  - mapped in CVA2sch to genericcode <Agency><LongName>
- listName=
  - mapped in CVA2sch to genericcode first <LongName>
- listVersionID=
  - mapped in CVA2sch to genericcode <Version>
- listURI=
  - mapped in CVA2sch to genericcode <LocationUri>
- listSchemeURI=
  - mapped in CVA2sch to genericcode <CanonicalVersionUri>

UBL-defined identifier element meta data attributes summarized from page 43

- schemeAgencyID=
  - mapped in CVA2sch to genericcode <Agency><Identifier>
- schemeAgencyName=
  - mapped in CVA2sch to genericcode <Agency><LongName>
- schemeName=
  - mapped in CVA2sch to genericcode first <LongName>
- schemeVersionID=
  - mapped in CVA2sch to genericcode <Version>
- schemeDataURI=
  - mapped in CVA2sch to genericcode <LocationUri>
- schemeURI=
  - mapped in CVA2sch to genericcode <CanonicalVersionUri>

## Polaris controlled-vocabulary declarations

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



Polaris U.K. Limited <http://www.polaris-uk.co.uk/> is owned and directed by the UK insurance industry

- assists in the development and adoption of business and technology standards to improve customer service for member organizations
- big users and definers of code lists within their industry

W3C schema-declared enumerations of code list-based information item data types

- instance-level meta data is built into and inseparable from the code list value
  - note how the code list value "B75 BS" incorporates an identifier for the code list base "B75" (for payment means) from which the value is obtained
  - this inhibits some strategies (described later) for extensibility and list evolution

```

01 <xs:element name="MethodOfPaymentCode" minOccurs="0" maxOccurs="1">
02 ...
03 <xs:complexType>
04 <xs:sequence>
05 <xs:element name="Value" minOccurs="1" maxOccurs="1">
06 ...
07 <xs:simpleType>
08 <xs:restriction base="xs:string">
09 <xs:enumeration value="B75 BS">
10 <xs:annotation>
11 <xs:documentation>Broker Statement</xs:documentation>
12 </xs:annotation>
13 </xs:enumeration>
14 <xs:enumeration value="B75 CC">
15 <xs:annotation>
16 <xs:documentation>Credit Card</xs:documentation>
17 </xs:annotation>
18 </xs:enumeration>
19 <xs:enumeration value="B75 DC">
20 <xs:annotation>
21 <xs:documentation>Debit Card</xs:documentation>
22 </xs:annotation>
23 </xs:enumeration>
24 ...
25 </xs:restriction>
26 </xs:simpleType>
27 </xs:element>
28 ...
29 </xs:sequence>
30 </xs:complexType>
31 </xs:element>

```

## Polaris controlled-vocabulary declarations (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



Multiple declarations of the same code list

- three copies of the declaration of TitleCode in MsgSchemaOfficeQuoteNBAt.xsd
- note in all three cases that the list of five values is a subset of a list of 88 possible values defined for all possible uses in documents

```

01 <xs:element name="IndividualName" minOccurs="0" maxOccurs="1">
02 ...
03 <xs:complexType>
04 <xs:sequence>
05 <xs:element name="TitleCode" minOccurs="0" maxOccurs="1">
06 ...
07 <xs:complexType>
08 <xs:sequence>
09 <xs:element name="Value" minOccurs="1" maxOccurs="1">
10 ...
11 <xs:simpleType>
12 <xs:restriction base="xs:string">
13 <xs:enumeration value="B53 001">
14 <xs:annotation>
15 <xs:documentation>Doctor</xs:documentation>
16 </xs:annotation>
17 </xs:enumeration>
18 <xs:enumeration value="B53 002">
19 <xs:annotation>
20 <xs:documentation>Miss</xs:documentation>
21 </xs:annotation>
22 </xs:enumeration>
23 <xs:enumeration value="B53 003">
24 <xs:annotation>
25 <xs:documentation>Mr</xs:documentation>
26 </xs:annotation>
27 </xs:enumeration>
28 <xs:enumeration value="B53 004">
29 <xs:annotation>
30 <xs:documentation>Mrs</xs:documentation>
31 </xs:annotation>
32 </xs:enumeration>
33 <xs:enumeration value="B53 005">
34 <xs:annotation>
35 <xs:documentation>Ms</xs:documentation>
36 </xs:annotation>
37 </xs:enumeration>
38 </xs:restriction>
39 </xs:simpleType>
40 </xs:element>
41 ...

```

## Polaris controlled-vocabulary declarations (cont.)

Chapter 2 - Defining and using controlled vocabularies  
Section 3 - Declarations of controlled vocabularies



The code list is maintained in a database (displayed here using two columns):

01 B53 040 Admiral	B53 056 Lieutenant General
02 B53 039 Air Chief Marshall	B53 021 Lord
03 B53 041 Air Commodore	B53 022 Major
04 B53 043 Air Marshall	B53 085 Major General
05 B53 044 Air Vice Marshall	B53 023 Master
06 B53 042 Airman	B53 064 Master Sergeant
07 B53 037 An t'Uasal	B53 002 Miss
08 B53 068 Baron	B53 036 Mother
09 B53 069 Baroness	B53 003 Mr
10 B53 038 Bean	B53 079 Mr Justice
11 B53 045 Bombardier	B53 004 Mrs
12 B53 008 Brigadier	B53 005 Ms
13 B53 027 Brother	B53 080 Prince
14 B53 046 Canon	B53 057 Private
15 B53 009 Captain	B53 024 Professor
16 B53 070 Cardinal	B53 077 Provost
17 B53 010 Chief	B53 029 Rabbi
18 B53 011 Colonel	B53 058 Rear Admiral
19 B53 012 Commander	B53 059 Regimental Sergeant Major
20 B53 013 Commodore	B53 006 Reverend
21 B53 047 Corporal	B53 032 Right Reverend
22 B53 028 Councillor	B53 060 Sergeant
23 B53 071 Count	B53 081 Sheikh
24 B53 072 Countess	B53 007 Sir
25 B53 014 Dame	B53 030 Sister
26 B53 048 Dean	B53 025 Squadron Leader
27 B53 001 Doctor	B53 061 Staff Sergeant
28 B53 087 Estate Of	B53 083 The Duke of
29 B53 034 Executor(s) of	B53 082 The Earl of
30 B53 035 Father	B53 073 The Honourable Lady
31 B53 049 Field Marshall	B53 086 The Honourable Miss
32 B53 015 Flight Lieutenant	B53 074 The Honourable Mrs
33 B53 050 Flight Sergeant	B53 075 The Honourable Sir
34 B53 016 General	B53 084 The Marquess of
35 B53 051 Group Captain	B53 066 The Most Reverend
36 B53 052 Gunner	B53 031 The Right Honourable
37 B53 017 Honourable	B53 067 The Venerable
38 B53 018 Judge	B53 088 Trustees Of
39 B53 019 Lady	B53 033 Very Reverend
40 B53 053 Lance Bombardier	B53 065 Viscount
41 B53 055 Lance Corporal	B53 078 Viscountess
42 B53 020 Lieutenant	B53 062 Warrant Officer 1
43 B53 054 Lieutenant Colonel	B53 063 Warrant Officer 2
44 B53 076 Lieutenant Commander	B53 026 Wing Commander

## Chapter 3 - Declaring controlled vocabularies



- Introduction - Declaring controlled vocabularies
- Section 1 - OASIS Genericcode 1.0
- Section 2 - OASIS Context/value association using genericcode
- Section 3 - Rendering controlled vocabularies

## Outcomes

- consider the different ways of declaring the enumerated values of controlled vocabularies
- understand the use and structure of genericcode files
- create new code lists where none existed before
- exploit existing controlled vocabularies when making new controlled vocabularies
- learn how to apply stylesheets to controlled vocabulary expressions

## Declaring controlled vocabularies

Introduction - Chapter 3 - Declaring controlled vocabularies



Standards are in development for the non-schema-based representation of a list of coded values

- trading partners may wish to trim or augment the list of coded values acceptable
- trading partners may wish to use different controlled vocabularies for a given information item found in different document contexts
- the representation of individual coded values includes documentary information and metadata
  - for detailed value description
  - for long-term maintenance and understanding
- OASIS genericode 1.0
  - <http://docs.oasis-open.org/codelist/genericode>
  - an XML representation standardized by the OASIS Code List Representation Technical Committee
  - <http://www.oasis-open.org/committees/codelist/>
  - "Defining an XML format for interchange, documentation and management of code lists (a.k.a. controlled vocabularies or coded value enumerations) in any processing context"
    - not obliged to use XML format *inside* the application
    - very common to compile the XML interchange format into an internal processing format
      - e.g. conversion to XSLT
      - e.g. implementation in database stored procedures
    - XML is designed for interchange and is not always conveniently structured for real-time processing

One could use schema enumerations but ...

- too inflexible for globally-defined information items
  - cannot have different sets of values in different document contexts for a globally-defined information item
- modifying the schemas means using non-standardized schema expressions
  - not bad in and of itself but requires extra assurances for compatibility
  - structural and lexical validation is assured if the standardized schema expressions are treated as read-only

Meta-data-only code list are important as placeholders

- effectively an infinite set of all possible codes satisfying the lexical rules
- indicating that a particular information item's value is from a controlled vocabulary but that there is no controlled vocabulary listing a set of codes
- e.g. only 18 of 91 UBL code lists are published with values, 73 uniquely-categorized code lists have only meta data
- users have the option of restricting the infinite list into a finite list

## OASIS Genericode 1.0

Chapter 3 - Declaring controlled vocabularies  
Section 1 - OASIS Genericode 1.0

An XML vocabulary for the representation of a set of coded values

- <http://docs.oasis-open.org/codelist/genericode>
- contributed to the formation of the OASIS Code List Representation TC
  - <http://www.oasis-open.org/committees/codelist/>
- also in use by other standardization committees e.g. FpML
- as of this writing the current version of genericode is 1.0
  - the code lists delivered in UBL 2.0 use genericode 0.4
  - the code lists delivered in UBL 2.0 Update use genericode 1.0
  - the committee is working on genericode version 1.1

Three main sections:

- code list-level meta data
  - defined by the element <Identification>
- available value-level meta data for coded values
  - defined by the element <ColumnSet>
- coded value enumeration and associated value-level meta data
  - defined by the element <SimpleCodeList>

Enumeration columns have at a minimum a key column for the coded value

- defined within <ColumnSet> by the element <Key> pointing to the appropriate <Column>
- other columns are available for meta data for the coded value

Enumeration rows have at a minimum a key value defining the coded value

- each <Row> defines the information for a coded value
- the <Value> corresponding to the key column defines the coded value
- the <Value> for other columns defines the meta data

Example: currency codes

- only code and name
  - the name is used as a description
- no other value-level meta data

Example: shipping container sizes and types

- documented properties:
  - container size name detail
  - length, width and height in millimeters
  - length, width and height in imperial
  - container type
- 30Mb of genericode XML



## OASIS Genericode 1.0 (cont.)

Chapter 3 - Declaring controlled vocabularies  
Section 1 - OASIS Genericode 1.0



## Excerpt from

<http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/cefact/CurrencyCode-2.0.gc>  
with simple item-level meta data:

- note the rows are limited to only code and name

```
01 <gc:CodeList
02 xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/">
03   <Identification>
04     <ShortName>CurrencyCode</ShortName>
05     <LongName xml:lang="en">Currency</LongName>
06     <LongName Identifier="listID">ISO 4217 Alpha</LongName>
07     <Version>2001</Version>
08     <CanonicalUri>urn:un:unece:uncefact:codelist:specification:54217
09 </CanonicalUri>
10     <CanonicalVersionUri>urn:un:unece:uncefact:codelist:
11 specification:54217:2001</CanonicalVersionUri>
12     <LocationUri>http://docs.oasis-open.org/ubl/os-ubl-2.0-update/
13 cl/gc/cefact/CurrencyCode-2.0.gc</LocationUri>
14     <Agency>
15       <LongName xml:lang="en">United Nations Economic Commission for
16 Europe</LongName>
17       <Identifier>6</Identifier>
18     ...
19     <ColumnSet>
20       <Column Id="code" Use="required">
21         ...
22         <Key Id="codeKey">
23           <ShortName>CodeKey</ShortName>
24           <ColumnRef Ref="code"/>
25         ...
26       <SimpleCodeList>
27         <Row>
28           <Value ColumnRef="code">
29             <SimpleValue>AED</SimpleValue>
30           </Value>
31           <Value ColumnRef="name">
32             <SimpleValue>Dirham</SimpleValue>
33           </Value>
34         </Row>
35         <Row>
36           <Value ColumnRef="code">
37             <SimpleValue>AFN</SimpleValue>
38           </Value>
39           <Value ColumnRef="name">
40             <SimpleValue>Afghani</SimpleValue>
41           </Value>
42         ...
43       </SimpleCodeList>
44     </ColumnSet>
45   </Identification>
46 </gc:CodeList>
```

## OASIS Genericode 1.0 (cont.)

Chapter 3 - Declaring controlled vocabularies  
Section 1 - OASIS Genericode 1.0



## Excerpt from

<http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/special-purpose/ContainerSizeCodeType-2.0.gc>  
with detailed item-level meta data:

- note the rows have far more information than just code and name, thus providing detailed meta data for each coded value
- the name is the synthesis of the other seven meta data properties

```
01 <gc:CodeList
02 xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/">
03   <Identification>
04     <ShortName xml:lang="en">Container Size Type</ShortName>
05     ...
06     <LongName xml:lang="en">
07       >Ship-planning Message Design Group</LongName>
08     ...
09   <SimpleCodeList>
10     <Row>
11       <Value ColumnRef="code"><SimpleValue>10GP</SimpleValue></Value>
12       <Value ColumnRef="name">
13         <SimpleValue>Unventilated general purpose container: 2991mm
14           (10') x 2438mm (8') x 2 438mm (8'0")</SimpleValue>
15       </Value>
16       <Value ColumnRef="lengthmm"><SimpleValue>2991</SimpleValue>...
17       <Value ColumnRef="lengthimperial"><SimpleValue>10'</SimpleValue>...
18       <Value ColumnRef="widthmm"><SimpleValue>2438</SimpleValue>...
19       <Value ColumnRef="widthimperial"><SimpleValue>8'</SimpleValue>...
20       <Value ColumnRef="heightmm"><SimpleValue>2 438</SimpleValue>...
21       <Value ColumnRef="heightimperial"><SimpleValue>8'0"</SimpleValue>...
22       <Value ColumnRef="type">
23         <SimpleValue>Unventilated general purpose container</SimpleValue>
24       </Value>
25     </Row>
26     <Row>
27       <Value ColumnRef="code"><SimpleValue>10G0</SimpleValue></Value>
28       <Value ColumnRef="name">
29         <SimpleValue>Unventilated general purpose container - Openings
30 at one or both ends: 2991mm (10') x 2438mm (8') x 2 438mm (8'0")
31 </SimpleValue>
32       </Value>
33       <Value ColumnRef="lengthmm"><SimpleValue>2991</SimpleValue>...
34       <Value ColumnRef="lengthimperial"><SimpleValue>10'</SimpleValue>...
35       <Value ColumnRef="widthmm"><SimpleValue>2438</SimpleValue>...
36     </Row>
37   </SimpleCodeList>
38 </gc:CodeList>
```

## OASIS Genericode 1.0 (cont.)

Chapter 3 - Declaring controlled vocabularies  
Section 1 - OASIS Genericode 1.0



Genericode files are validated using an W3C Schema expression of constraints

- `http://docs.oasis-open.org/codelist/genericode/xsd/genericode.xsd`
- includes `http://docs.oasis-open.org/codelist/genericode/xsd/xml.xsd`

The development of genericode version 1.0 published in 2007

- the work is continuing by the OASIS technical committee
- backward-compatible version 1.1 being developed to package multiple code lists

## Example code list validation scenario

Chapter 3 - Declaring controlled vocabularies  
Section 1 - OASIS Genericode 1.0



Two trading partners agree to exchange an order

- one is in Canada and the other is in the US
- the buyer may be either Canadian- or US-based
  - this limits the country sub-entity code of the buyer to be either provinces or states
- the seller can only be US-based
  - this limits the country sub-entity code of the seller to be only states
- the payment means available supplements the UBL values for payment means
  - all of the UBL-standardized values can be used
  - the value "SHP" can be used to represent "payment by the exchange of sheep"
- currency codes must be limited to either US dollars or Canadian dollars



## Specifying a controlled vocabulary

Chapter 3 - Declaring controlled vocabularies  
Section 1 - OASIS Genericcode 1.0



Example genericcode file enumerating Canadian provinces:

- artefacts/Crane-artfacts/Crane-CVA2sch/scenario/CA\_CountrySubentityCode.gc
- note that this is version "2" of the list of Canadian provinces and territories
  - the semantic behind the code "NT" representing "Northwest Territories" changed April 1, 1999 when the new territory "Nunavut" (new code "NU") was created from the split of the old territory into two
    - trivia: Nunavut has a land area of 2 million square kilometers (770,000 square miles; larger than the state of Texas or the country of France) and a total population of 28,000 people
- important note: this genericcode file is for illustration purposes and is not intended to be an official representation of the information

```
01 <gc:CodeList
02 xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
03   <Identification>
04     <ShortName>provinces</ShortName>
05     <LongName>Canadian Provinces</LongName>
06     <Version>2</Version>
07     ...
08   </Identification>
09   ...
10   <SimpleCodeList>
11     <Row>
12       <Value ColumnRef="code">
13         <SimpleValue>AB</SimpleValue>
14       </Value>
15       <Value ColumnRef="name">
16         <SimpleValue>Alberta</SimpleValue>
17       </Value>
18     </Row>
19     <Row>
20       <Value ColumnRef="code">
21         <SimpleValue>BC</SimpleValue>
22       </Value>
23       <Value ColumnRef="name">
24         <SimpleValue>British Columbia</SimpleValue>
25       </Value>
26     </Row>
27     ...
```

## Specifying a controlled vocabulary (cont.)

Chapter 3 - Declaring controlled vocabularies  
Section 1 - OASIS Genericcode 1.0



Example genericcode file enumerating US states:

- artefacts/Crane-artfacts/Crane-CVA2sch/scenario/US\_CountrySubentityCode.gc
- important note: this genericcode file is for illustration purposes and is not intended to be an official representation of the information

```
01 <gc:CodeList
02 xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
03   <Identification>
04     <ShortName>states</ShortName>
05     <LongName>US States</LongName>
06     <Version>1</Version>
07     ...
08   </Identification>
09   ...
10   <SimpleCodeList>
11     <Row>
12       <Value ColumnRef="code">
13         <SimpleValue>AL</SimpleValue>
14       </Value>
15       <Value ColumnRef="name">
16         <SimpleValue>ALABAMA</SimpleValue>
17       </Value>
18     </Row>
19     <Row>
20       <Value ColumnRef="code">
21         <SimpleValue>AK</SimpleValue>
22       </Value>
23       <Value ColumnRef="name">
24         <SimpleValue>ALASKA</SimpleValue>
25       </Value>
26     </Row>
27     ...
```

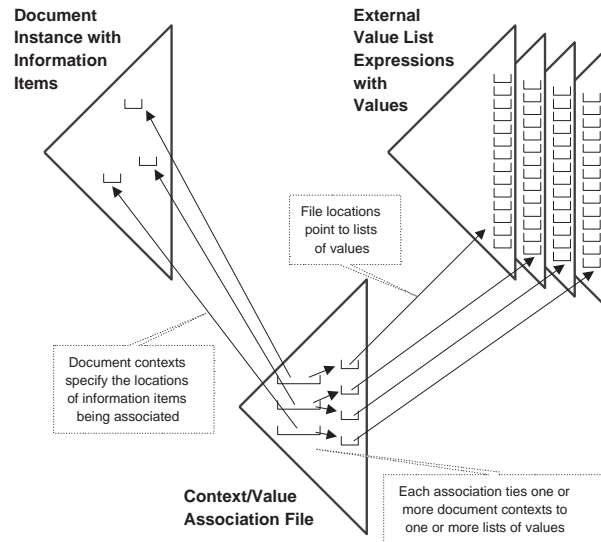
## OASIS Context/value association using genericode

Chapter 3 - Declaring controlled vocabularies  
Section 2 - OASIS Context/value association using genericode



### Extension and restriction accomplished using association

- declarative approach to the association of a controlled vocabulary to an XML document context
- specifies lists of codes by pointing to external genericode files
- pointing to more than one external genericode file specifies the union of the files



### No implication of use or applicability or process

- useful in data entry user interface applications
- useful in document validation

## Extending a controlled vocabulary

Chapter 3 - Declaring controlled vocabularies  
Section 2 - OASIS Context/value association using genericode



### The intuition to pick up an existing list and simply add values to it is misguided

- the new list is a different list than the original list so no longer be considered the original list and must, therefore, be identified distinctly with different list-level meta data

### List-level meta data impacts on the business validity of instances

- the new expanded list is obliged to have different list-level meta data so as not to be interpreted as a bona fide original list
- instance-level-meta-data-qualified specifications from the original list will not validate when the modified list's list-level meta data has been modified

### Extending an item's values to beyond the existing set of codes involves simultaneously referencing the original list and any lists of extended values

- association marries the multiple lists together into one suite of values allowed for a given information item
- specifications of unique values amongst the merged values need not specify list meta data but may choose to do so
- specifications of ambiguous values amongst the merged values are obliged to specify list meta data to disambiguate from which list the value is obtained

### Permanently growing a list involves publishing versions

- list-level meta data distinguishes one set of values from another set of values
- the semantics of the codes are described for a particular version of the growing list
  - semantics can change over time, so the list version is important to distinguish the true meaning of the values
- during migration association can reference both versions as applicable
  - can provide for the union of older and newer versions for any given information item

## Extending a controlled vocabulary (cont.)

Chapter 3 - Declaring controlled vocabularies  
Section 2 - OASIS Context/value association using genericcode



### Example genericcode file enumerating a single value:

- artefacts/Crane-artifacts/Crane-CVA2sch/scenario/Additional\_PaymentMeansCode.gc

```

01 <gc:CodeList
02 xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
03   <Identification>
04     <ShortName>AdditionalPaymentMeansCode</ShortName>
05     <LongName xml:lang="en">Additional Payment Means</LongName>
06     <Version>1</Version>
07     <CanonicalUri>urn:x-company:PaymentMeansCode</CanonicalUri>
08     <CanonicalVersionUri>urn:x-company:PaymentMeansCode-1.0
09   </CanonicalVersionUri>
10   </Identification>
11   <ColumnSet>
12     <Column Id="code" Use="required">
13       <ShortName>Code</ShortName>
14       <Data Type="normalizedString"/>
15     </Column>
16     <Column Id="name" Use="optional">
17       <ShortName>Name</ShortName>
18       <Data Type="string"/>
19     </Column>
20     <Key Id="codeKey">
21       <ShortName>CodeKey</ShortName>
22       <ColumnRef Ref="code"/>
23     </Key>
24   </ColumnSet>
25   <SimpleCodeList>
26     <Row>
27       <Value ColumnRef="code">
28         <SimpleValue>SHP</SimpleValue>
29       </Value>
30       <Value ColumnRef="name">
31         <SimpleValue>Exchange of Sheep</SimpleValue>
32       </Value>
33     </Row>
34   </SimpleCodeList>
35 </gc:CodeList>

```

Of note:

- the list-level meta data does not look like that of the UN-ECE Payment Means

## Restricting a controlled vocabulary

Chapter 3 - Declaring controlled vocabularies  
Section 2 - OASIS Context/value association using genericcode



The intuition to pick up an existing list and simply remove values from it is misguided

- but the new list must be identified with different list-level meta data
  - the new list is a different list than the original list so no longer be considered the original list so original meta data cannot apply

List-level meta data impacts on the validity of instances

- the new reduced list is obliged to have different meta data so as not to be interpreted as a bona fide original list
- instance-level-meta-data-qualified specifications from the original list will not validate if the list-level meta data has been modified

Introduces the concept of the "masquerade"

- restricting an items values to a subset of an existing set of codes involves referencing the new list of reduced values while masquerading the reduced list to be the original list
- association provides the opportunity to masquerade the reduced list to appear as the original list

## Restricting a controlled vocabulary (cont.)

Chapter 3 - Declaring controlled vocabularies  
Section 2 - OASIS Context/value association using genericcode



Example genericcode file limiting an already-constrained schema enumeration for currencies:

- artefacts/Crane-artifacts/Crane-CVA2sch/scenario/CAUS\_CurrencyCode.gc
- compare this list's meta data with the original list shown on page 56

```

01 <gc:CodeList
02 xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
03   <Identification>
04     <ShortName>CAUSCurrencyCode</ShortName>
05     <LongName>Canadian and US Currency Codes</LongName>
06     <Version>1</Version>
07     <CanonicalUri>urn:x-company:CAUS-currency</CanonicalUri>
08     <CanonicalVersionUri>urn:x-company:CAUS-currency:1
09   </CanonicalVersionUri>
10   </Identification>
11   ...
12   <SimpleCodeList>
13     <Row>
14       <Value ColumnRef="code">
15         <SimpleValue>CAD</SimpleValue>
16       </Value>
17       <Value ColumnRef="name">
18         <SimpleValue>Canadian Dollar</SimpleValue>
19       </Value>
20     </Row>
21     <Row>
22       <Value ColumnRef="code">
23         <SimpleValue>USD</SimpleValue>
24       </Value>
25       <Value ColumnRef="name">
26         <SimpleValue>US Dollar</SimpleValue>
27       </Value>
28     </Row>
29   </SimpleCodeList>
30 </gc:CodeList>

```

Of note:

- the list-level meta data does not look like that of the UN-ECE Currency Codes

## Rendering controlled vocabularies

Chapter 3 - Declaring controlled vocabularies  
Section 3 - Rendering controlled vocabularies



Some audiences do not appreciate having to read raw XML to interpret the contents

- angle brackets are distracting
- the volume of XML markup overwhelms the information content of the instance

Crane Softwrights Ltd. has published free developer resources with which to render a genericcode code expression to HTML

- XSLT 1.0 stylesheet: Crane-genericcode2html.xsl
- standalone production of HTML from genericcode file
- browser-based viewing of HTML from genericcode file

## Standalone production of an HTML rendering

Chapter 3 - Declaring controlled vocabularies  
Section 3 - Rendering controlled vocabularies



Consider the HTML rendering of the earlier example of a genericcode file in page 64 rendered using:

```
- java -jar saxon.jar -o Additional_PaymentMeansCode.html
  Additional_PaymentMeansCode.gc Crane-genericcode2html.xsl
```

The resulting HTML file when rendered appears as follows:

List	
ShortName = AdditionalPaymentMeansCode LongName (xml:lang="en") = Additional Payment Means Version = 1 CanonicalUri = urn:x-company:PaymentMeansCode CanonicalVersionUri = urn:x-company:PaymentMeansCode-1.0	
Code <small><i>Id="code"</i> (required) Type="normalizedString" [Key Id="codeKey": CodeKey]</small>	Name <small><i>Id="name"</i> (optional) Type="string"</small>
SHP	Exchange of Sheep

## Browser-based viewing of an HTML rendering

Chapter 3 - Declaring controlled vocabularies  
Section 3 - Rendering controlled vocabularies



W3C XML stylesheet association standardized `xml-stylesheet` processing instruction:

- `http://www.w3.org/1999/06/REC-xml-stylesheet-19990629`
- `href` = points to the stylesheet file (modify as required)
- `type="text/xsl"` used to indicate the interpretation of the stylesheet
- in Windows drag the file from Windows Explorer to the browser canvas to render

The modified file includes the processing instruction recognized by XML processing tools:

- examples in the `samp/ss` directory

```
01 <?xml version="1.0" encoding="ISO-8859-1"?>
02 <?xml-stylesheet type="text/xsl" href="Crane-genericcode2html.xsl"?>
03 <gc:CodeList
04 xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
05   <Identification>
06     <ShortName>AdditionalPaymentMeansCode</ShortName>
07     <LongName xml:lang="en">Additional Payment Means</LongName>
08     <Version>1</Version>
09     <CanonicalUri>urn:x-company:PaymentMeansCode</CanonicalUri>
10     <CanonicalVersionUri>urn:x-company:PaymentMeansCode-1.0
11   </CanonicalVersionUri>
12   </Identification>
13   <ColumnSet>
14     <Column Id="code" Use="required">
15       <ShortName>Code</ShortName>
16       <Data Type="normalizedString" />
17     </Column>
18     <Column Id="name" Use="optional">
19       <ShortName>Name</ShortName>
20       <Data Type="string" />
21     </Column>
22     <Key Id="codeKey">
23       <ShortName>CodeKey</ShortName>
24       <ColumnRef Ref="code" />
25     </Key>
26   </ColumnSet>
27   <SimpleCodeList>
28     <Row>
29       <Value ColumnRef="code">
30         <SimpleValue>SHP</SimpleValue>
31       </Value>
32       <Value ColumnRef="name">
33         <SimpleValue>Exchange of Sheep</SimpleValue>
34       </Value>
35     </Row>
36   </SimpleCodeList>
37 </gc:CodeList>
```

## Problems triggering browser-based rendering

Chapter 3 - Declaring controlled vocabularies  
Section 3 - Rendering controlled vocabularies



It is possible that the rendering initially shows up without interpretation:

- Internet Explorer does not recognize the file extension to be an XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Crane-genericcode2html.xsl"?>
<gc:codeList xmlns:gc="http://docs.oasis-open.org/codeList/ns/genericcodeList" >
  <Identification>
    <ShortName>AdditionalPaymentMeansCode</ShortName>
    <LongName>Additional Payment Means</LongName>
    <Version>1</Version>
    <CanonicalUri>urn:x-company:PaymentMeansCode</CanonicalUri>
    <CanonicalVersionUri>urn:x-company:PaymentMeansCode-1.0</CanonicalVersionUri>
  </Identification>
  <ColumnSet>
    <Column Id="code" Use="required">
      <ShortName>Code</ShortName>
      <Data Type="normalizedString"/>
    </Column>
    <Column Id="name" Use="optional">
      <ShortName>Name</ShortName>
      <Data Type="string"/>
    </Column>
    <Key Id="codeKey">
      <ShortName>CodeKey</ShortName>
      <ColumnRef Ref="code"/>
    </Key>
  </ColumnSet>
  <SimpleCodeList>
    <Row>
      <value columnRef="code">
        <Simplevalue>SHP</Simplevalue>
      </value>
      <value columnRef="name">
        <Simplevalue>Exchange of sheep</Simplevalue>
      </value>
    </Row>
  </SimpleCodeList>
</gc:codeList>
```

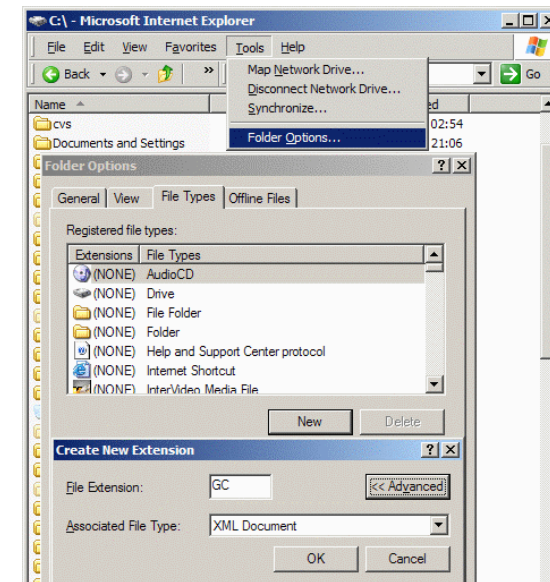
## Problems triggering browser-based rendering (cont.)

Chapter 3 - Declaring controlled vocabularies  
Section 3 - Rendering controlled vocabularies



One can teach Windows XP to get the effect shown in page 68:

- it is possible to indicate that any given file extension is an XML document
- from either the Control Panel or a Windows Explorer "Tools" menu, select "Folder Options"
- go to the "File Types" tab
- press the "New" button to define a new registered file type
- enter the file extension where indicated
- press the "Advanced >>" button
- select "XML Document" for the "Associated File Type"



## Chapter 4 - Controlled vocabulary representation detail



- Introduction - Controlled vocabulary representation detail
- Section 1 - Genericcode vocabulary
- Section 2 - Mapping genericcode meta data to XML instances

## Controlled vocabulary representation detail

Introduction - Chapter 4 - Controlled vocabulary representation detail



### Genericcode

- an XML vocabulary for the representation of an enumerated set of values
- <http://docs.oasis-open.org/codelist/genericcode/xsd/genericcode.xsd>
  - base vocabulary for the genericcode namespace
- <http://docs.oasis-open.org/codelist/genericcode/xsd/xml.xsd>
  - imported vocabulary for the XML namespace in support of `xml:lang=`

### Meta data identification of an enumerated set of values

- information regarding identifying the list as a whole
  - names by which the list is known
  - version of the particular list
  - resource identifiers with which to retrieve the list contents
- information regarding identifying the custodian
  - agency
- information regarding each individual value

### Many features for supporting sets of code lists or derived code lists

- Schematron-based context/value association stylesheets as delivered support only simple single-key code lists in standalone declarations
  - interpreting multiple keys or a set of sets or a derivation of sets is not supported
- this documentation is limited to the available document types and to the supported constructs only of simple single-key standalone declarations
  - this documentation is not meant to be exhaustive documentation for genericcode files
  - please see the OASIS code list representation committee deliverables for more details on genericcode files

## Genericode information

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericode vocabulary



### Three entry points into the information

- a genericode file can validly be one of three kinds of collections of information
- code list set
  - the definition of an aggregate collection of code lists or other sets of enumerated values
  - includes meta data for the set of sets
  - this kind of genericode file is not referenced by context/value association files
  - no other information regarding code list sets is included in this documentation
- column set
  - the definition of a group of meta data columns for enumerated values
    - distinct from the meta data used for identifying code lists
  - this is useful when sharing a comment set of column definitions across multiple code list files
    - allows a code list to use an outboard definition of columns rather than an embedded definition of columns
  - this kind of genericode file is not referenced by context/value association files
    - the methodology assumes the column definitions are embedded within the code list files in a specified column set
    - no other information regarding column sets is included in this documentation
- code list
  - the definition of a single enumerated set of values
  - includes meta data for the single set
  - includes meta data for individual values in the set
  - only this kind of simple genericode file with simple single-key enumeration sets is used by context/value association files
  - no other information regarding derived code lists is included in this documentation

Recall the example UBL enumeration for payment means on page 44

- an example of a simple genericode file with a simple single-key enumeration

## Genericode list-level meta data

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericode vocabulary



### Information regarding identifying the list as a whole

- names by which the list is known
  - short name (token)
  - long names (normalized strings)
    - optional distinguishing meta data can be used for each name
      - arbitrary distinction between one long name and other long names
        - Identifier="listID" attribute indicates the long name referenced with UN/CEFACT listID= meta data
        - other long name is the list name
      - the language in which the long name is written
  - canonical name (URI string)
- distinct revision of the particular list
  - version (token)
  - canonical version (URI string)
- addresses with which to retrieve the list contents
  - resource location identifiers (URI strings)

### Information regarding identifying the custodian

- agency short name (token)
- agency long names (normalized strings)
  - optional distinguishing meta data can be used for each name
- agency identifiers (normalized strings)
  - optional distinguishing meta data can be used for each identifier



## Genericcode standalone simple enumeration sets

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



An enumeration set is defined by columns and rows

- one row per member of the enumeration set
  - there may be no rows, thus the file defines a meta-data-only code list
- the columns contain the meta data for each member
  - there must be a column set, but it may be empty
  - following typical ISO use, the empty lists found in UBL define candidate columns of "Code" and "Name" for meta-data-only code lists, but these are not normative and have no meaning as there are no rows
  - context/value association stylesheets do not support a reference to an external definition of columns for a column set and will only respect embedded specifications of columns in a column set

A single column or a combination of columns uniquely identifies the key for each row

- by definition the keys must be unique so as to be unambiguous
- context/value association stylesheets assume a key is only a single column
  - when not named, the first declared key column is used as the key
- other uses of genericcode may support multiple key columns

Meta data (columns) defined for each enumeration member (a row)

- an arbitrary amount of user-defined information can be kept for members
- meta data may be specified differently for every enumeration in the set
- choice of whether the meta data item is mandatory or optional
- specification of the data type of the meta data item
- simple values or richly-marked up elements, attributes and mixed content

## Genericcode XML

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



Only one entry point expected for those files pointed to from context/value association files

- a standalone specification of a simple code list

`<gc:CodeList>`

- the specification of a standalone enumerated set of values
- this element is in the genericcode namespace
- `xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/"`
  - any prefix can be used, the "gc" prefix is not reserved in any way
  - unusually, the namespace applies only to the document element, not the descendants
  - UBL 2.0 uses a pre-release version of genericcode with the namespace URI `"xmlns:gc="http://genericcode.org/2006/ns/CodeList/0.4/"`
  - UBL 2.0 update package republished all code lists using the finalized URI
- all descendent elements of the genericcode vocabulary are in no namespace
- mandatory `<Identification>` child and `<ColumnSet>` child
- optional `<SimpleCodeList>` child
  - not used if the file specifies a meta-data-only enumeration set

All elements other than the document element are in no namespace

- thus, it is most convenient to put the document element in a prefixed namespace rather than the default namespace, and to keep the default namespace undefined

## Genericcode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



### <Identification>

- (mandatory) name, version, resource and custodian meta data for the list

### <ShortName>

- (mandatory) a token (no spaces) naming the enumeration set suitable for software artefacts
- `xml:lang=`
  - optional language specification of the derived language

### <LongName>

- (optional and repeatable) a normalized string
- `Identifier=`
  - (optional) used to distinguish between multiple names
  - `Identifier="listID"`
    - recognized by the methodology stylesheets as a UN/CEFACT list identifier
- `xml:lang=`
  - (optional) language specification of the derived language

## Genericcode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



### <Identification> (cont.)

#### <Version>

- (mandatory) the version of the enumeration set

#### <CanonicalUri>

- (mandatory) a unique identifier independent of the version (i.e. for all versions) of the enumeration set

#### <CanonicalVersionUri>

- (mandatory) a unique identifier for this particular version of the enumeration set

#### <LocationUri>

- (optional and repeatable) a suggested location for a genericcode expression of this enumeration set

#### <AlternateFormatLocationUri>

- (optional and repeatable) suggested locations for non-genericcode expressions of this enumeration set

## Genericcode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



## &lt;Identification&gt; (cont.)

## &lt;Agency&gt;

- (optional) information regarding the custodian of the enumeration set
- <ShortName>
  - (optional) a token (no spaces) naming the custodian suitable for software artefacts
  - xml:lang=
    - optional language specification of the derived language
  - not used by context/value association stylesheets
- <LongName>
  - (optional and repeatable) a normalized string
  - Identifier=
    - (optional) used to distinguish between multiple names
  - xml:lang=
    - (optional) language specification of the derived language
- <Identifier>
  - (optional and repeatable) a normalized string identifier
  - Identifier=
    - (optional) used to distinguish between multiple identifiers
  - xml:lang=
    - (optional) language specification of the derived language

## Genericcode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



## &lt;ColumnSet&gt;

- (mandatory) identification of the meta data columns for the enumeration set rows
- DefaultDatatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
  - (optional) the URI of the data type library utilized by the column type specifications
  - by default the W3C Schema data types are used
  - while other values are possible, context/value association stylesheets do not inspect this attribute

Omitted column set for the example UBL enumeration set shown on page 44:

```

01  <ColumnSet>
02    <Column Id="code" Use="required">
03      <ShortName>Code</ShortName>
04      <Data Type="normalizedString"/>
05    </Column>
06    <Column Id="name" Use="optional">
07      <ShortName>Name</ShortName>
08      <Data Type="string"/>
09    </Column>
10    <Key Id="codeKey">
11      <ShortName>CodeKey</ShortName>
12      <ColumnRef Ref="code"/>
13    </Key>
14  </ColumnSet>

```

The context/value association stylesheets only support single-valued keys

- the genericcode specification allows keys to be combined in a multi-valued lookup
- context/value association key= provides for selecting which one key to use by the key's id= attribute

## Genericcode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



## &lt;ColumnSet&gt; (cont.)

## &lt;Column&gt;

- (mandatory and repeatable) information about a piece of value-level meta data used for each enumeration in the set
- Id=
  - (mandatory) identification of the column for referencing purposes
- Use="optional" or Use="required"
  - (mandatory) specification of the row's required specification of the column data
- <Annotation>
  - (optional and repeatable) user information (not used by the context/value association stylesheets)
- <ShortName>
  - (mandatory) a token (no spaces) naming the column suitable for software artefacts
  - xml:lang=
    - optional language specification of the derived language
- <LongName>
  - (optional and repeatable) a normalized string
    - Identifier=
      - used to distinguish between multiple names
    - xml:lang=
      - optional language specification of the derived language
- <CanonicalUri>
  - (optional) a unique identifier independent of the version (i.e. for all versions) of the column
- <CanonicalVersionUri>
  - (optional) a unique identifier for this particular version of the column
- <Data>
  - (mandatory) specification of the data type for the column
  - Type=
    - (mandatory) the data type from the data type library
  - DatatypeLibrary=
    - (optional) the data type library overriding the default data type library
  - <Annotation>
    - (optional) user information (not used by context/value association stylesheets)
  - <Parameter>
    - (optional and repeatable) facet parameter (not used by context/value association stylesheets)
    - (mandatory) ShortName= for the facet
    - (optional) LongName= for the facet

## Genericcode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



## &lt;ColumnSet&gt; (cont.)

## &lt;Key&gt;

- (mandatory and repeatable) definition of a key column of unique values used for enumeration values
- Id=
  - (mandatory) identification of the key for referencing purposes
- <Annotation>
  - (optional and repeatable) user information (not used by context/value association stylesheets)
- <ShortName>
  - (mandatory) a token (no spaces) naming the column suitable for software artefacts
  - xml:lang=
    - optional language specification of the derived language
- <LongName>
  - (optional and repeatable) a normalized string
    - Identifier=
      - used to distinguish between multiple names
    - xml:lang=
      - optional language specification of the derived language
- <CanonicalUri>
  - (optional) a unique identifier independent of the version (i.e. for all versions) of the key
- <CanonicalVersionUri>
  - (optional) a unique identifier for this particular version of the key
- <ColumnRef>
  - (mandatory) specification of the key column
  - Ref=
    - (mandatory) the identifier of the key column
  - <Annotation>
    - (optional) user information (not used by context/value association stylesheets)
    - <Description>
      - (optional) human-readable information
    - <AppInfo>
      - (optional) machine-readable information

## Genericode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericode vocabulary



### <SimpleCodeList>

- this wraps all of the enumeration member rows of the enumeration set
- this element is not used when there are no value constraints for the list
  - the list is not empty, it just has no constraints
  - the list is effectively an infinite set of all possible values meeting the lexical constraints

### <Annotation>

- (optional) user information (not used by context/value association stylesheets)
- <Description>
  - (optional) human-readable information
- <AppInfo>
  - (optional) machine-readable information

## Genericode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericode vocabulary



### <SimpleCodeList> (cont.)

#### <Row>

- (optional and repeatable) specification of a member of the enumeration set
- <Annotation>
  - (optional) user information (not used by context/value association stylesheets)
  - <Description>
    - (optional) human-readable information
  - <AppInfo>
    - (optional) machine-readable information
- <Value>
  - (required) specification of the value for this meta data column for this enumeration row
  - the order of <Value> elements is irrelevant
  - ColumnRef=
    - (required) pointer to the identifier of the column of meta data this value specified (allows arbitrary order of <Value> elements)
  - <Annotation>
    - (optional) user information (not used by context/value association stylesheets)
    - <Description>
      - (optional) human-readable information
    - <AppInfo>
      - (optional) machine-readable information
  - <SimpleValue>
    - optional in the model but required by context/value association stylesheets
    - this is mutually-exclusive with <ComplexValue>
  - <ComplexValue>
    - any foreign content (non-genericode vocabulary) can be used for this value
    - this is mutually-exclusive with <SimpleValue>
    - this information is not used by context/value association stylesheets

## Genericcode XML (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 1 - Genericcode vocabulary



Entry points to genericcode files not pointed to by context/value association

- these entry points are defined by genericcode but not supported for context/value association files
- no other information regarding these is included in this documentation

`<gc:CodeListSet>`

- document type for the definition of an aggregate collection of code lists
- this element is in the genericcode namespace
  - all descendent elements of the genericcode vocabulary are in no namespace

`<gc:ColumnSet>`

- document type for the definition of a group of enumeration set meta data columns and/or keys
- this element is in the genericcode namespace
  - all descendent elements of the genericcode vocabulary are in no namespace

## Mapping genericcode meta data to XML instances

Chapter 4 - Controlled vocabulary representation detail  
Section 2 - Mapping genericcode meta data to XML instances



One must consider instance-level meta data when designing an XML vocabulary

- if an information item is allowed to have a code or an identifier, it should also have associated instance-level meta data
- allows the author of the XML to qualify the code with list-level meta data values in the instance-level meta data properties

Each aspect of list-level meta data should be allowed to be overridden

- should try to cover all aspects of genericcode identification list-level facets

Crane validation stylesheets are modular supporting adaptation for any instance-level meta data

- one can create a `Crane-ABC-genericcode2Schematron.xml` stylesheet using a `Crane-ABC-Metadata.xml` fragment that uses values from genericcode code lists checking any instance-level meta data in the ABC vocabulary
- stylesheets include a pro-forma configuration supporting no instance-level meta data
  - the `Crane-NM-genericcode2Schematron.xml` stylesheet uses values from genericcode code lists without checking any instance-level meta data
  - meta data stub stylesheet is `Crane-NM-Metadata.xml`
- stylesheets include a configuration supporting UBL instance-level meta data
  - the `Crane-UBL-genericcode2Schematron.xml` stylesheet uses values from genericcode code lists checking any instance-level meta data
  - meta data stylesheet is `Crane-UBL-Metadata.xml`

## Mapping genericode meta data to XML instances (cont.)

Chapter 4 - Controlled vocabulary representation detail  
Section 2 - Mapping genericode meta data to XML instances



Recall the available UN/CEFACT meta data attributes for UBL information items on page 48

- the following lists map UBL information item meta data attributes (on the left) to the genericode meta data information items (on the right)

The following mappings are for UN/CEFACT-defined supplementary components:

- currencyCodeListVersionID= for currencyID= maps to Version
- unitCodeListVersionID= for unitCode= maps to Version
- unitCodeListID= for unitCode= maps to LongName[@Identifier='listID'] or LongName[1]
- unitCodeListAgencyID= for unitCode= maps to Agency/Identifier
- unitCodeListAgencyName= for unitCode= maps to Agency/LongName

The following mappings are for UBL-defined code list element meta data (those elements with names ending in "Code"):

- listID= maps to LongName[@Identifier='listID'] or LongName[1]
- listAgencyID= maps to Agency/Identifier
- listAgencyName= maps to Agency/LongName
- listName= maps to LongName[1]
- listVersionID= maps to Version
- listURI= maps to LocationUri
- listSchemeURI= maps to CanonicalVersionUri

The following mappings are for UBL-defined identifier element meta data (those elements with names ending in "ID"):

- schemeAgencyID= maps to Agency/Identifier
- schemeAgencyName= maps to Agency/LongName
- schemeName= maps to LongName[1]
- schemeVersionID= maps to Version
- schemeDataURI= maps to LocationUri
- schemeURI= maps to CanonicalVersionUri

## Chapter 5 - Associating controlled vocabularies in XML documents



- Introduction - Constraining information items using controlled vocabularies
- Section 1 - CVA for validation using Schematron
- Section 2 - Code list methodology example
- Section 3 - Rendering context/value association files

### Outcomes

- consider the requirements of associating information items defined by controlled vocabularies
- overview the published methodology used by the UBL technical committee
- walk through an extended example of the validation of controlled vocabularies
- learn how to apply stylesheets to context/value expressions

## Constraining information items using controlled vocabularies

Introduction - Chapter 5 - Associating controlled vocabularies in XML documents



Three kinds of constraints to be validated for an XML document

- structural constraints ensure information items are correctly found
- lexical constraints ensure information items are correctly formed
- value constraints ensure information items are correctly understood

Constraining the document structure and lexical patterns is independent of business/value rules

- a community of users can publish an agreed upon schema to validate information items are correctly found and formed

Constraining information item use of controlled vocabularies is very dependent on business/value rules

- business/value rules implied by the nature of the information item
  - e.g. points of a compass will never change
- business/value rules imposed by a community of users
  - e.g. the document status codes for the condition of a document in a transaction
- business/value rules agreed upon between trading partners
  - e.g. identification of account numbers for particular purposes

Typical use of W3C Schema conflates structural and value constraints inflexibly

- one gets more flexibility by separating value constraints from structural constraints
- only structural constraints should be imposed across a community of users
  - standard should constrain how the information is found and how it is formed, not how it is valued
  - very infrequent changes to the structure of information being interchanged
  - changes imply big impacts on applications and processing
- value constraints should be selectively imposed
  - changes in trading partners
  - changes in business practices over time
  - possibly frequent changes to the values allowed by different parties
  - once programs accommodate a given set of values, changing the subsets of values in use doesn't change the applications
- business rules should be selectively added
  - private requirements could never be anticipated by standards committees

## Context/value association

Introduction - Chapter 5 - Associating controlled vocabularies in XML documents



Context/value association files

- [http://www.oasis-open.org/committees/document.php?document\\_id=29990](http://www.oasis-open.org/committees/document.php?document_id=29990)
- an XML vocabulary for associating document contexts with specified values
- suitable for constraining document entry in a user interface
- suitable for document validation before application processing
- techniques for specifying, restricting and extending lists for the purposes of validation

Masquerading meta data when restricting a large list to a subset of values

- the validation needs to match an instance's use of large list meta data to a declaration of a subset list using subset list list-level meta data
- the subset list list-level meta data necessarily is different than the list-level meta data of the list from which it is derived
- the subset list masquerades as the list from which it is derived so that instance-level meta data doesn't use the custom subset list list-level meta data

ISO/IEC 19757-3 Schematron deployment

- as supplied, the methodology reports context/value constraint violations in simple text
- Schematron can alternatively be deployed with different available reporting techniques



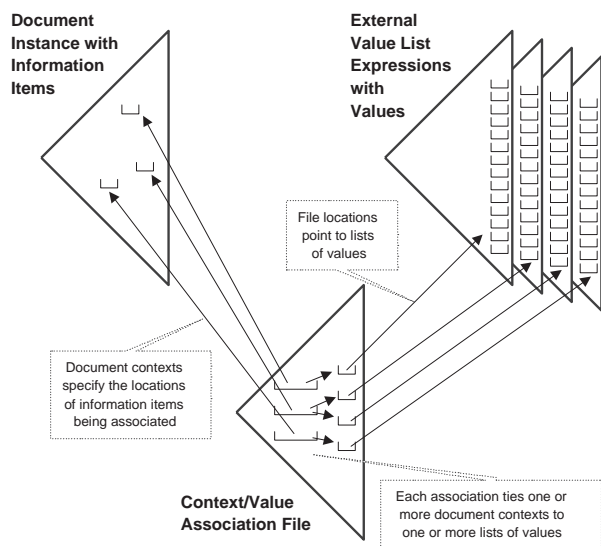
## Context/value association (cont.)

Introduction - Chapter 5 - Associating controlled vocabularies in XML documents



The principles of context/value association are as follows:

- XML documents have information items that need to be validated
  - the locations (contexts) of those items can be addressed using XPath addresses
- genericcode files have values and list meta data to use for validation
  - the locations of those files can be declared with URL addresses
  - the identity of each list is uniquely specified in order to be referenced multiple times
- an association marries a document context with a set of genericcode files
  - each XPath document context is specified with the identities of the genericcode declarations
- validation checks values found in document contexts against genericcode files linked by the association for the document context
  - any present meta data in the document context is checked with the available genericcode meta data



## Context/value association (cont.)

Introduction - Chapter 5 - Associating controlled vocabularies in XML documents



Appropriate for constraining data entry application user interfaces

- used as a front end to a user preventing the data entry of different values
  - drop-down lists
  - radio buttons
  - check boxes
- the end result of editing an instance is that the values are all from the associated lists
- the value-level meta data can be presented to the user
  - assists the user in choosing which value or values to use
- the options to include instance-level meta data should be offered
  - reflects the list-level meta data for the list from where the values are taken

Appropriate for constraining data validation

- used as a front end to an application that implements the logic for all possible values
- selective association for business scenarios prevents the application from acting on inappropriate values for a given transaction
  - relationships between specific partners may be different
  - different profiles of using documents may constrain particular values

Only the CVA vocabulary is standardized by OASIS, not how it is used

- the file format and the semantics represented by the elements and attributes are being standardized by OASIS
- any implementation is considered out of scope of the committee work

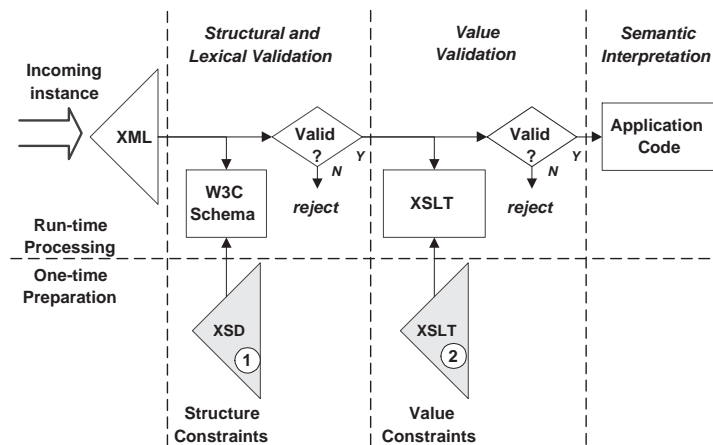
## Using context/value association for validation

Introduction - Chapter 5 - Associating controlled vocabularies in XML documents



Separates structural/lexical validation from value validation

- an XML document is checked using a two-step process
- the first pass for structural and lexical validation passes
- the second pass reports that a coded value used for a currency is unexpected
- the document structure and lexical content can be constrained by standardization
  - e.g. the UBL technical committee publishes normative W3C schemas
- the document controlled-value content is constrained by business requirements between trading partners
  - e.g. the UBL committee publishes default coded value checks
    - defaultCodeList.xsl
  - trading partners can use this value validation methodology to create their own value checking second-pass process



Document arrives at application unchanged

- validation only confirms the use of structure and content, without modifying it

Second pass results meaningless without first pass being successful

- the values must be correctly found and correctly formed before checking the actual values produces an accurate result

## Using context/value association for validation (cont.)

Introduction - Chapter 5 - Associating controlled vocabularies in XML documents



Crane-CVA2sch package from Crane Softwrights Ltd. web site

- historically developed in the OASIS UBL Technical Committee
- moved into the OASIS Code List Representation Technical Committee
- moved out of the OASIS Code List Representation Technical Committee
  - the committee decided to focus on file formats and not methodologies
  - intellectual property returned to Crane Softwrights Ltd.
- Crane is donating CVA2sch to an Apache Schematron project

A methodology for code list and value validation based on ISO/IEC 19757-3 Schematron

- an information item is asserted to have one of an allowed set of predetermined values
  - a failed assertion is a value validation error
- assertions are derived from context/value associations

Schematron is usually implemented using the Extensible Stylesheet Language (XSLT)

- the supplied Schematron stylesheet for stylesheets is a copy of the publicly-available reference XSLT implementation
  - <http://www.schematron.com>
- the methodology supplies a wrapper stylesheet for the reference skeleton
- other non-XSLT implementations of Schematron exist
  - e.g. Amara/Scimitar implements ISO Schematron in Python
    - <http://uche.ogbuji.net:8080/uche.ogbuji.net/tech/4Suite/amara/>
    - same architecture as reference XSLT implementation in that Scimitar is a Python program that writes a Python program that performs the validation

The XSLT generated to implement the Schematron assertions is used as the second pass of validation to test XML instances for having correct controlled-vocabulary values

- the testing relies on the first-pass structural validation, having already confirmed the structure and lexical values used in the instance
- without the first pass confirming the accurate presence of information items, the second pass is meaningless

The methodology supports the incorporation of any number of sets of Schematron assertions

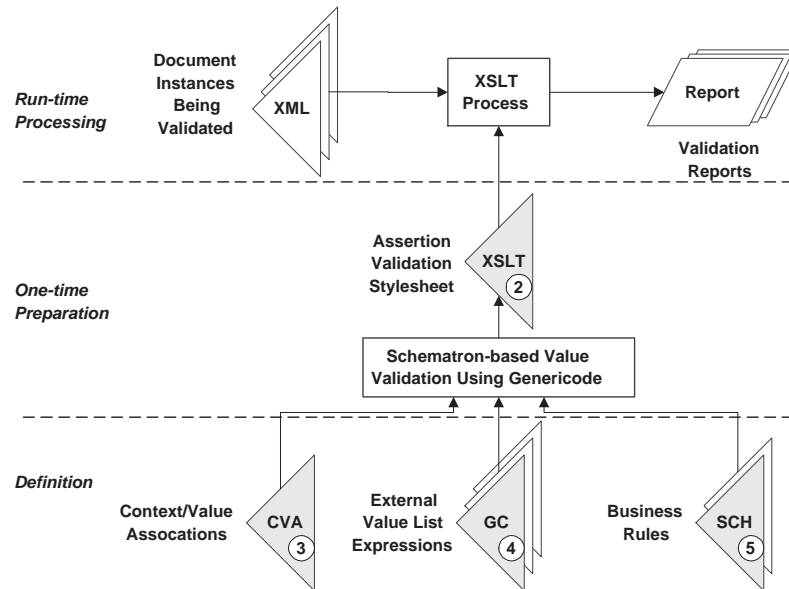
- ISO Schematron supports the inclusion of multiple schema fragments into a single schema expression
- business rules related or unrelated to code lists may be expressed as Schematron assertions
  - the trading partner schema can then include business rules in addition to coded value rules

## Using context/value association for validation (cont.)

Introduction - Chapter 5 - Associating controlled vocabularies in XML documents



Overview of the process to prepare the second pass value validation XSLT stylesheet:



- the circled labels in the diagram are indicated by the parenthesized numbers
- the inputs:
  - (3) the specification of contexts uses the context/value association XML vocabulary defined by the OASIS Code List Representation TC
  - (4) the specification of coded values uses the genericcode vocabulary defined by the OASIS Code List Representation TC
  - (5) supplemental business rules are specified using ISO/IEC 19757-3 Schematron
- the output:
  - (2) an XSLT stylesheet (or some other implementation of Schematron assertion checking)

Recall Validating controlled vocabularies (page 24)

- the XSLT created here (2) plugs in to the two-step validation process

Recall Context/value association (page 92)

- all three documents on that diagram are shown here as instances being validated, the context value association files and the external value list expressions

## CVA for validation using Schematron

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron

The document structure and lexical constraints are already formalized as usable artefacts

- document schemas are the artefacts published by committees and communities to be used in the structure/lexical validation process
- a UBL customization is the published set of W3C schemas for a user community
  - subsets the standardized business objects to requirements included in UBL
  - extends the standardized business objects for requirements not included in UBL

The objective is to formalize the implementation guide between trading partners as usable artefacts

- "context/value association files" are the artefacts published by committees, communities and trading partners to be used in the business/value validation process
- an XML document expresses the association between document contexts and their associated controlled values
- the files of declared agreed-upon requirements form part of the trading partner agreement

The formalized artefacts are exchanged and transformed into a programmatic script

- the requirements for business/value validation found in the context/value association file XML vocabulary are converted into an executable Schematron schema
- the Schematron schema implements the business/value validation as an executable process
  - each trading partner may have a different implementation of Schematron based on their own system criteria
    - e.g. XSLT 1 or XSLT 2 stylesheets
    - e.g. Python programming language

Stylesheets can make the information in the formalized artefacts presentable

- the context/value association file XML can be transformed into HTML for rendering purposes

## CVA for validation using Schematron (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



Two or three inputs into the methodology's processes (shown on page 25)

- expressing the association of document contexts and coded values ("3")
  - using both absolute and relative XPath expressions, one can specify information items in many or in individual contexts
  - e.g. all contextual uses of `<cbc:DocumentCurrencyCode>` will use a limited set of currency values
  - the values used in UBL 2.0 are enumerated in the support package
    - not included in the UBL 2.0 specification package
- expressing the coded values allowed to be used in possible contexts ("4")
  - using genericcode, one can specify sets of predetermined values and their meta data
    - e.g. the set of currencies for countries engaged in cross-border transactions is limited to only the values "CAD" (Canadian Dollar) and "USD" (United States Dollar)
  - the UBL TC supplies the values used in UBL 2.0 for default validation for conformance to an arbitrary set of code lists
    - found in the <http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/> directory
- optionally expressing business rules related or unrelated to predetermined values ("5")
  - using Schematron, one can specify arbitrary business rules about the content found in information items
    - e.g. "the total value of the invoice cannot exceed \$10,000 in the currency of the transaction"
  - the UBL TC has not yet published any business rules for UBL documents
    - business rules are typically the purview of trading partners
    - customization designers may choose to specify business rules for arbitrary constraints
      - e.g. absence of empty elements
      - e.g. string length of particular values

One output of the methodology's processes

- the second-pass Schematron value validation (e.g. XSLT stylesheet) ("2")
  - the UBL TC supplies a stylesheet used for UBL 2.0 default values:
    - delivered as  
<http://docs.oasis-open.org/ubl/os-UBL-2.0/val/defaultCodeList.xsl>

## Document context representation

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



XPath expressions are used to specify the context of information items in an XML document

- `@attribute-name` or rarely `@namespace-prefix:attribute-name`
  - note that no attribute names in UBL are namespace-qualified
- `namespace-prefix:element-name` or simply `element-name`
  - note that all element names in UBL are namespace-qualified
- `parent/child` or `parent/@attribute`
  - used when the child is an immediate descendant of the parent
- `ancestor/descendant` or `ancestor/@attribute`
  - used when the descendant is at any depth below the ancestor
- `/document-element` absolute address
  - used to indicate the top of the XML document tree, starting a fully-qualified path to the item being addressed
- one cannot readily determine available contexts just by looking at schemata
  - a schema declaration shows only one level of children when following the UBL NDR

## Document context representation (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



## Absolute (anchored) XPath address:

- the following are addressed completely from the document element to the target document context

```
01 /po:Order/cac:TaxTotal/cbc:TaxAmount/@currencyID
02 /po:Order/cbc:DocumentCurrencyCode
03 /po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/
04                                     cbc:CountrySubentityCode
05 /po:Order/cac:SellerSupplierParty/cac:Party/cac:Address/
06                                     cbc:CountrySubentityCode
```

## Relative (unanchored) XPath address without context:

- the following information items can occur in any context in the document

```
01 @currencyID
02 cbc:DocumentCurrencyCode
03 cbc:CountrySubentityCode
```

## Relative (unanchored) XPath address with "narrow" context:

- the following are addressing particular document contexts below an apex

```
01 cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:CountrySubentityCode
02 cac:SellerSupplierParty/cac:Party/cac:Address/cbc:CountrySubentityCode
```

## Relative (unanchored) XPath address with "wide" context:

- the following are addressing all document contexts below an apex

```
01 cac:BuyerCustomerParty//cbc:CountrySubentityCode
02 cac:SellerSupplierParty//cbc:CountrySubentityCode
```

## Document context representation (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



## Summary of contexts for UBL documents, distilled from UBL document models:

- delivered as part of the future UBL support package
- contexts are minimally unique
  - the indicates paths are as short as they can be while still being unique contexts
  - the XPath addresses in the context files have no injected white-space

```
01 Invoice: (58 uses of code list data types; 58 unique parents;
02         4580 unique contexts)
03
04 cbc:AccountTypeCodeType: (1 unique parent; 20 contexts;
05                           extends udt:CodeType)
06   cbc:AccountTypeCode (20 contexts)
07     cac:InvoiceLine/cac:Price/cac:AllowanceCharge/cac:PaymentMeans/
08                           cac:PayeeFinancialAccount/cbc:AccountTypeCode
09     cac:InvoiceLine/cac:Price/cac:AllowanceCharge/cac:PaymentMeans/
10                           cac:PayerFinancialAccount/cbc:AccountTypeCode
11     ...
12   in:Invoice/cac:AllowanceCharge/cac:PaymentMeans/
13                           cac:PayeeFinancialAccount/cbc:AccountTypeCode
14   in:Invoice/cac:AllowanceCharge/cac:PaymentMeans/
15                           cac:PayerFinancialAccount/cbc:AccountTypeCode
16   ...
17
18 cbc:CountrySubentityCodeType: (1 unique parent; 370 contexts;
19                               extends udt:CodeType)
20   cbc:CountrySubentityCode (370 contexts)
21   ...
22   cac:BuyerCustomerParty/cac:Party/cac:AgentParty/
23                           cac:PhysicalLocation/cbc:CountrySubentityCode
24   cac:BuyerCustomerParty/cac:Party/
25                           cac:PhysicalLocation/cbc:CountrySubentityCode
26   ...
27   cac:SellerSupplierParty/cac:Party/cac:AgentParty/
28                           cac:PhysicalLocation/cbc:CountrySubentityCode
29   cac:SellerSupplierParty/cac:Party/
30                           cac:PhysicalLocation/cbc:CountrySubentityCode
31   ...
```

## In the above the report shows:

- in all of UBL there is only one parent of cbc:AccountTypeCodeType, that being cbc:AccountTypeCode and that there are 20 unique contexts in which cbc:AccountTypeCode is being used
- there is only one parent of cbc:CountrySubentityCodeType, that being cbc:CountrySubentityCode and that there are 370 unique contexts in which cbc:CountrySubentityCode is being used

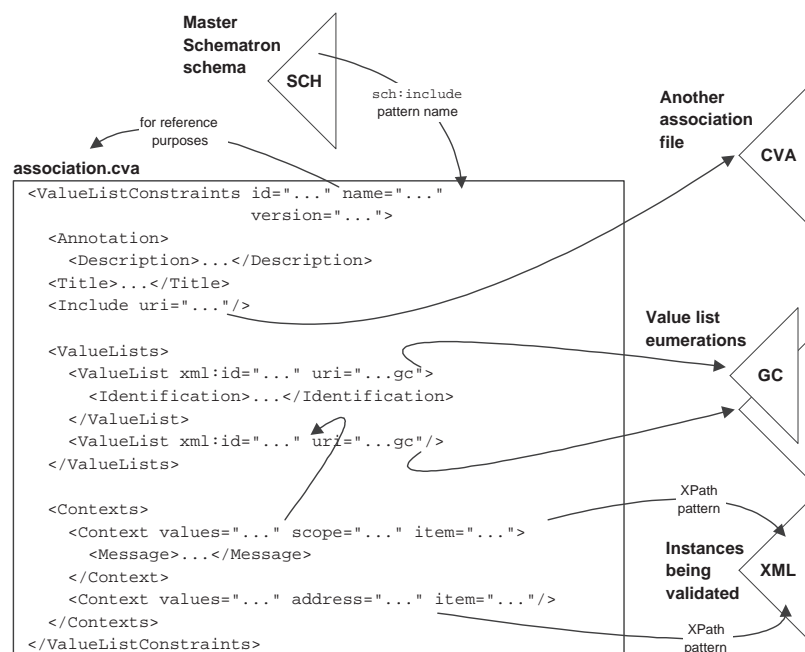
## Context/value association specification

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



A code list context association file associates values with document contexts

- the methodology transforms an association file into a Schematron pattern
  - suitable to be included into a master Schematron schema using `<sch:include>`
  - the Schematron pattern needs to be identified by a pattern name
- the `version=` attribute, and `<Annotation>/<Description>` and `<Title>` elements are used only for documentary purposes
- an association file may include other association files when generating the pattern
  - the `<Include>` directive brings in other contexts and code lists
- value lists are identified within an association file by an identifier using `<ValueList>`
  - the `uri=` points to the actual code list enumeration genericcode file
- contexts refer to document information items by their XPath addressees using `<Context>`
  - the `values=` points to one or more value lists by their internal identifiers



These files can be used as part of a formal trading partner agreement

- unambiguous specification of value constraints

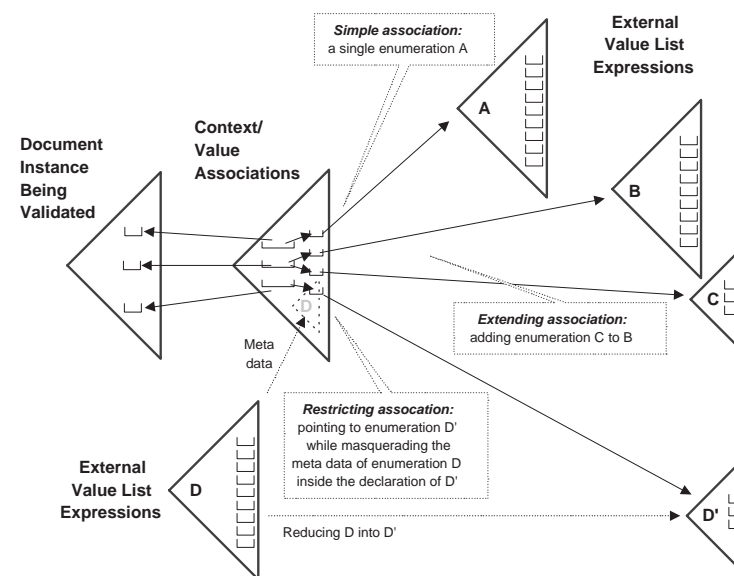
## Association specification, extension and restriction

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



The association implements simple, extended or restricted value lists

- the diagram illustrates three associations, one of each kind
- simple enumeration association: A
  - the one association points to a single external value list
- extending enumeration association: B extended by C
  - the one association simultaneously points to both B and C value lists
  - similarly, one association can point to the union of multiple unrelated lists if the combined set of values is required
- restricting enumeration association: D' restriction of D
  - the one association points to the reduced value list D'
  - the reduced list is treated as the complete list by masquerading D' as the complete list D
  - the meta data of D is embedded inside of the value list declaration for D'



## Association specification, extension and restriction (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron

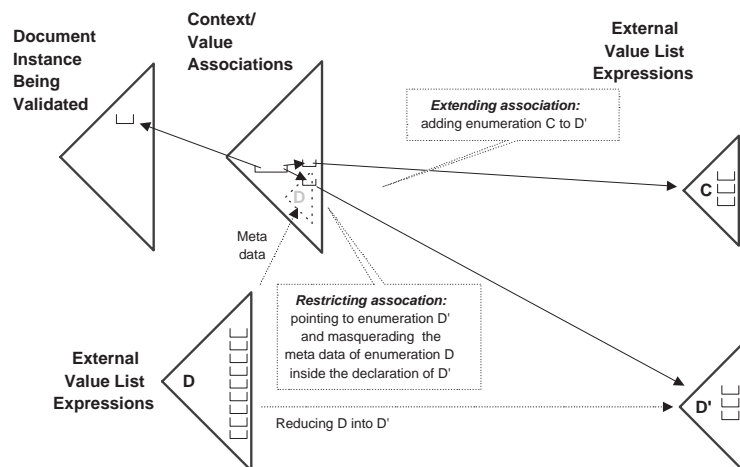


Consider the situation where one wishes to associate for a given document context the combination of a restriction of another published list plus a set of extra codes not included in the published list:

- the diagram illustrates extending a restricted list
- restricting enumeration association: D' restriction of D
  - the one association points to the reduced value list D'
  - the reduced list is treated as the complete list by masquerading D' as the complete list D
  - the meta data of D is embedded inside of the value list declaration for D'
- extending enumeration association: D' extended by C
  - the one association simultaneously points to both D' and C value lists

As a real-world example, consider transportation status codes:

- UBL has published a set of transportation status codes
  - (in the drawing this would be list D)
- a user doesn't want to use all of the published codes, so they create their own restricted list
  - (in the drawing this would be list D')
- the user has a set of additional status codes not published by UBL
  - (in the drawing this would be list C)
- the association combines the subset list D' of published codes plus the extended list C of additional codes



## Context/value pro forma associations

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



Pro forma example of context associations specified across all documents:

- recall the diagram illustrating the relationships on page 102
- note how all of the contexts are specified with relative XPath addresses

```

01 <?xml version="1.0" encoding="US-ASCII"?>
02 <ValueListConstraints ... namespaces for XPath addresses ...
03 xmlns="http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/
04 cd2-1.0/"
05   id="urn:x-optional-unique-identifier-for-external-referencing"
06   name="required-unique-name-token-for-internal-referencing"
07   version="Revision: 27b - 2006-11-23 15:00z">
08   <Annotation>
09     <Description>
10       This is included for illustrative purposes.
11     </Description>
12   </Annotation>
13
14   <Title>
15     This is the main context/value association file for project X.
16   </Title>
17
18   <Include uri="other-assoc-file-1.cva">
19     <Annotation>
20       <Description>
21         Incorporating information from another file.
22       </Description>
23     </Annotation>
24   </Include>
25   <Include uri="other-assoc-file-2.cva">
26     <Annotation>
27       <Description>
28         Incorporating information from yet another file.
29       </Description>
30     </Annotation>
31   </Include>

```

(pro forma example continued on the next page)

The above identifies the context/value association file itself and the component pieces that are included

- the optional id= is used for identifying this file from outside of this file
  - perhaps in cataloguing artefacts used in a transaction
- the mandatory name= is used to name the Schematron pattern that is included into the master Schematron schema used for validation
- the version-, <Description> and <Title> are documentary



## Context/value pro forma associations (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



## Pro forma example (cont.)

- recall the diagram illustrating the relationships on page 102

```

01 <ValueLists>
02   <Annotation>
03     <Description>
04       All the lists available to be used.
05     </Description>
06   </Annotation>
07   <ValueList xml:id="a1" uri="enumeration1.gc">
08     <Annotation>
09       <Description>
10         A set of external values with no masquerading meta data.
11       </Description>
12     </Annotation>
13   </ValueList>
14   <ValueList xml:id="a2" uri="enumeration2.gc">
15     <Annotation>
16       <Description>
17         A set of external values with masquerading version meta data.
18       </Description>
19     </Annotation>
20     <Identification>
21       <Version>1.4</Version>
22     </Identification>
23   </ValueList>
24   <ValueList xml:id="a3" uri="enumeration3.gc">
25     <Annotation>
26       <Description>
27         Another set of external values with no masquerading meta data.
28       </Description>
29     </Annotation>
30   </ValueList>
31 </ValueLists>

```

## Context/value pro forma associations (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



## Pro forma example (cont.)

- recall the diagram illustrating the relationships on page 102

```

01 <Contexts>
02   <Annotation>
03     <Description>
04       All the contexts using lists.
05     </Description>
06   </Annotation>
07   <Context item="@item-a" values="a2">
08     <Annotation>
09       <Description>
10         Associating a set of values with all item-a= attributes.
11       </Description>
12     </Annotation>
13   </Context>
14   <Context item="item-b" scope="context-b1" values="a1 a3">
15     <Annotation>
16       <Description>
17         Associating two sets of values with all item-b descendants
18         of the context-b1 element.
19       </Description>
20     </Annotation>
21   </Context>
22   <Context item="item-b" address="context-b2/item-b"
23     values="a3" mark="characteristic-1">
24     <Annotation>
25       <Description>
26         Associating a set of values with item-b children of the
27         context-b2 element.
28       </Description>
29     </Annotation>
30   </Context>
31 </Contexts>

```



## Context/value pro forma associations (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



This example checks all `item-a` attributes regardless of document context

- also regardless of document type
- no use of `scope=` or `address=` attributes

Two different ways in this example to handle `item-b` attributes

- when a child of the `context-b2` element, the values of list `enumeration3.gc` apply
  - the use of `address=` is important if the `item-b` element has more descendent contexts of `context-b2` than just being a child
  - note the inclusion of the `item-b` name in the address
- when a descendant or child of the `context-b1` element, the values of both lists `enumeration1.gc` and `enumeration3.gc` apply
  - the use of `scope=` specifies all `item-b` elements in descendent contexts of `context-b1`
  - note the exclusion of the `item-b` name in the address
- the prioritized order of the last two implies that the `item-b` child of `context-b2` that is also a descendant of `context-b1` will be treated as a child of `context-b2` and not as a descendant of `context-b1`

## Context/value pro forma associations (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



Pro forma example of context associations specified for different documents:

- recall the diagram illustrating the relationships on page 102
- note the use of the absolute XPath addresses `/po:Order` and `/in:Invoice` in the contexts

```

01 <?xml version="1.0" encoding="US-ASCII"?>
02 <ValueListConstraints xmlns="http://docs.oasis-open.org/codelist/ns/
03 ContextValueAssociation/cd2-1.0/"
04   xmlns:in="urn:oasis:names:specification:ubl:schema:xsd:Invoice-1.0"
05   xmlns:po="urn:oasis:names:specification:ubl:schema:xsd:Order-1.0"
06   name="code-list-rules">
07   <ValueLists>
08     <ValueList xml:id="a1" uri="enumeration1.gc"/>
09     <ValueList xml:id="a2" uri="enumeration2.gc"/>
10     <ValueList xml:id="a3" uri="enumeration3.gc"/>
11   </ValueLists>
12   <Contexts>
13     <Context item="@item-a" values="a2"/>
14     <Context item="@item-b" scope="/po:Order//context-b1"
15       values="a1 a3"/>
16     <Context item="@item-b" scope="/in:Invoice//context-b1"
17       values="a3" mark="characteristic-1"/>
18   </Contexts>
19 </ValueListConstraints>

```

Two different ways in the example above to handle `item-b` elements:

- when a descendant of a `context-b1` element in a purchase order document two value lists apply
- when a descendant of a `context-b1` element in an invoice document only one value list applies

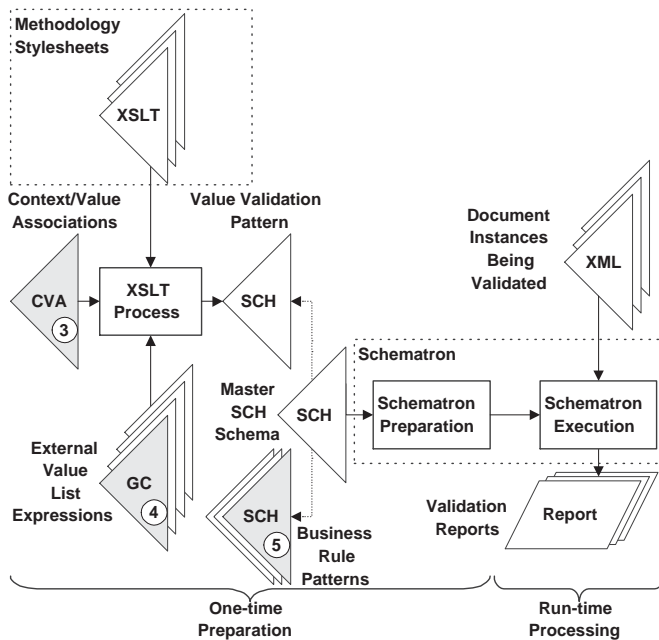
## Context/value validation implementation

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



Two separate implementations of running code are used to effect the result:

- the "Methodology Stylesheets" box represents the CVA XSLT stylesheets supplied with this methodology to transform context/value association files into a Schematron pattern
- the "Schematron" box represents the particular implementation of Schematron being deployed by a user of this methodology
  - the methodology package supplies an XSLT implementation of Schematron that exits with a non-zero return code when it reports violations to the standard error port in a simple text format
  - alternative implementations of Schematron are publicly-available from <http://www.Schematron.com> including versions that report violations in a rich XML vocabulary for subsequent downstream processing



Recall Validating controlled vocabularies (page 25)

Recall Validating controlled vocabularies (page 24)

## Context/value validation implementation (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 1 - CVA for validation using Schematron



Context/value associations and external code list expressions are converted to Schematron

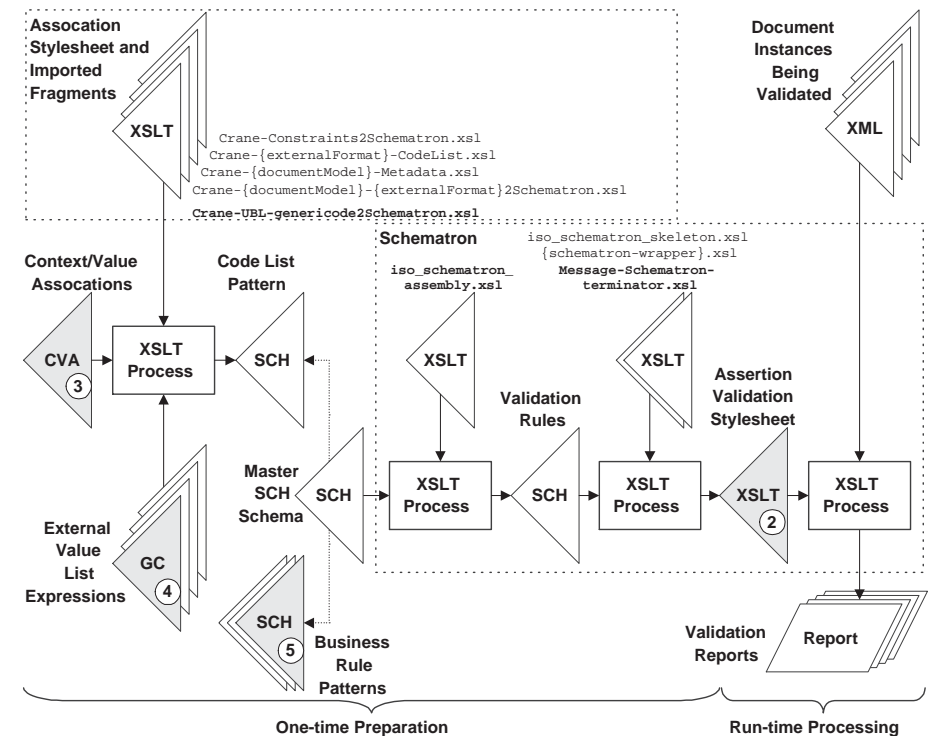
- Crane-UBL-genericcode2Schematron.xsl (UBL elements and meta data)
- Crane-NM-genericcode2Schematron.xsl (no meta data)

Schematron patterns assembled into a complete Schematron schema

- schematron-ISO-assembly.xsl

Schematron schema translated into an XSLT stylesheet

- Message-Schematron-terminator.xsl
- a wrapper of the iso\_schematron\_skeleton.xsl implementation



Recall Validating controlled vocabularies (page 25)

Recall Validating controlled vocabularies (page 24)

## Example code list validation scenarios

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



Recall the example scenario where two trading partners agree to exchange an order

- one is in Canada and the other is in the US
- currency codes must be limited to either US dollars or Canadian dollars
- the buyer may be either Canadian- or US-based
  - this limits the country sub-entity code of the buyer to be either provinces or states
- the seller can only be US-based
  - this limits the country sub-entity code of the seller to be only states
- the payment means supplements the available UBL values for payment means
  - all of the UBL-standardized values can be used
  - the value "SHP" can be used to represent "payment by the exchange of sheep"

A second (related) scenario includes a business rule

- same code list constraints, but the total value of the order must be less than \$10,000 of the order's currency
  - the `cbc:toBePaidAmount` for the order must be less than 10000.

The scenario/ directory included with the methodology documentation has the files illustrating the two scenarios

- instances of the UBL Order document to be validated for the use of coded values:
  - `order-test-good*.xml` and `order-test-bad*.xml`
- `codes-only-constraints.sch`
  - a Schematron schema that only checks for the coded values
- `total-constraints.sch`
  - a Schematron schema that checks for both the coded values and the business rule
- both Schematron schemas rely on an external Schematron pattern that isn't included
  - `order-constraints.sch` needs to be synthesized to test the coded value limitations

## Code list methodology example

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



Complete code list association file for the example scenario

- this associates the limited currency list and the province and state lists according to the requirements in the illustrative scenario

```

01 <?xml version="1.0" encoding="US-ASCII"?>
02 <ValueListConstraints xmlns="http://docs.oasis-open.org/codelist/ns/
03 ContextValueAssociation/cd2-1.0/"
04   xmlns:cbc="urn:oasis:...:CommonBasicComponents-2"
05   xmlns:cac="urn:oasis:...:CommonAggregateComponents-2"
06   xmlns:sch="http://purl.oclc.org/dsdl/schematron"
07   id="urn:x-illustration"
08   name="code-list-rules"
09   version="$Id: order-constraints.cva,v 1.8 2008/11/11 ...">
10
11 <Annotation>
12   <Description>
13     This is an illustrative example of all of the features of specifying
14     the
15     context/value constraints that one can express for XML documents.
16     The validation requirements for this contrived scenario are as
17     follows:
18       - the UN/CEFACT currency code list is restricted to be
19       - only Canadian and US dollars
20       - the seller must be in the US
21       - the buyer may be in either Canada or the US
22       - the definition for Payment Means is extended to include
23       - both UBL definitions and additional definitions
24   </Description>
25 </Annotation>
26
27 <Title>
28   Illustration of code list constraints - order-constraints.cva
29 </Title>

```

(association file continued on next page)

Of note above:

- the name of the pattern generated is "code-list-rules"
- the title information is meant to be terse
- the identification is meant to be simple text and terse
- the description information may be detailed and include rich markup (not shown in this example)

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



## Association file (cont.)

```

01  <!--list all of the genericcode expressions of agreed-upon code list
02    value enumerations-->
03  <ValueLists>
04    <ValueList xml:id="currency" uri="CAUS_CurrencyCode.gc">
05      <Annotation>
06        <Description>
07          Restricted to only Canadian and US dollars.
08        </Description>
09      </Annotation>
10      <Identification>
11        <ShortName>CurrencyCode</ShortName>
12        <LongName>Currency</LongName>
13        <LongName Identifier='listID'>ISO 4217 Alpha</LongName>
14        <Version>2001</Version>
15        <CanonicalUri>urn:un:unece:uncefact:codelist:specification:
16 54217</CanonicalUri>
17        <CanonicalVersionUri>urn:un:unece:uncefact:codelist:
18 specification:54217:2001</CanonicalVersionUri>
19        <Agency>
20          <LongName>United Nations Economic Commission for
21 Europe</LongName>
22          <Identifier>6</Identifier>
23        </Agency>
24      </Identification>
25    </ValueList>
26    <ValueList xml:id="states" uri="US_CountrySubentityCode.gc">
27      <Annotation>
28        <Description>
29          List of US states
30        </Description>
31      </Annotation>
32    </ValueList>
33    <ValueList xml:id="provinces" uri="CA_CountrySubentityCode.gc">
34      <Annotation>
35        <Description>
36          List of Canadian provinces
37        </Description>
38      </Annotation>
39    </ValueList>

```

(association file continued on next page)

Of note above:

- the declaration for the external restricted currency code list uses the masquerading meta data of the complete code list

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



## Association file (cont.)

```

01    <ValueList xml:id="tax-ids" uri="TaxIdentifier.gc" key="codeKey">
02      <Annotation>
03        <Description>
04          List of tax type identifiers
05        </Description>
06      </Annotation>
07    </ValueList>
08    <ValueList xml:id="payments" uri="UBL_PaymentMeansCode-2.0.gc">
09      <Annotation>
10        <Description>
11          Copied from the UBL 2.0 suite:
12          http://docs.oasis-open.org/ubl/cs-UBL-2.0/
13        </Description>
14      </Annotation>
15    </ValueList>
16    <ValueList xml:id="additional_payments"
17      uri="Additional_PaymentMeansCode.gc">
18      <Annotation>
19        <Description>
20          An extra set of possible payment means.
21        </Description>
22      </Annotation>
23    </ValueList>
24  </ValueLists>

```

(association file continued on next page)

Of note above:

- the UBL payment means code list is unchanged and used in its entirety
- an extended list of payment means codes is also declared ready to be used in the association declaration

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



## Association file (cont.)

```

01  <!--list all of the contexts in which the value enumerations are used;
02      where two or more contexts might match a given node in the input,
03      list them here in order of most-important to least important
match-->
04  <Contexts>
05    <Context item="@currencyID" values="currency">
06      <Annotation>
07        <Description>
08          All currencies are restricted to only Canadian and US dollars.
09        </Description>
10      </Annotation>
11    </Context>
12    <Context item="cbc:CountrySubentityCode"
13      scope="cac:BuyerCustomerParty"
14      values="provinces states">
15      <Annotation>
16        <Description>
17          The buyer can be in either Canada or the US.
18        </Description>
19      </Annotation>
20      <Message>Invalid province or state '<sch:value-of select="."/>'
21 for buyer "<sch:value-of select="ancestor::cac:BuyerCustomerParty/
22 cac:Party/cac:PartyName/cbc:Name"/>"</Message>
23    </Context>
24    <Context item="cbc:CountrySubentityCode"
25      scope="cac:SellerSupplierParty"
26      values="states">
27      <Annotation>
28        <Description>
29          The seller can only be in the US.
30        </Description>
31      </Annotation>
32      <Message>Invalid state '<sch:value-of select="."/>' for seller
33 "<sch:value-of select="ancestor::cac:SellerSupplierParty/cac:Party/
34 cac:PartyName/cbc:Name"/>"</Message>
35    </Context>

```

(association file continued on next page)

Of note above:

- the first context utilizes default violation reporting
- the second and third contexts specify a custom Schematron error message used in the reporting of the value violation

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



## Association file (cont.)

```

01    <Context item="cbc:ID" address="cac:TaxCategory/cbc:ID"
02      values="tax-ids">
03      <Annotation>
04        <Description>
05          Limit the recognized tax identifiers
06        </Description>
07      </Annotation>
08    </Context>
09    <Context item="cbc:PaymentMeansCode" mark="money"
10      values="payments additional_payments">
11      <Annotation>
12        <Description>
13          The payments can be by either standard or supplemental means.
14        </Description>
15      </Annotation>
16    </Context>
17  </Contexts>
18 </ValueListConstraints>

```

Of note above:

- the fourth and fifth contexts utilize default violation reporting
- the last association shows how the UBL payment means is extended by the additional payment means
- the last association is marked as being related to a money issue, perhaps distinguishing this in downstream processing as being a more or less important issue

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Sample master referencing only the validation of select code lists

- codes-only-constraints.sch
- used to pull in code lists
- no reference to other business rules
- note the need in this master schema to use Schematron declarations for the namespaces that are used within the included Schematron pattern
  - the methodology stylesheets that create the included pattern file incorporate a copy of these namespace declarations in the comments of the included file for ease of copy/paste into the master schema

```
01 <schema xmlns="http://purl.oclc.org/dsdl/schematron">
02 <title>Code list value assertions</title>
03 <ns prefix="cbc" uri="urn:oasis:...:CommonBasicComponents-2"/>
04 <ns prefix="cac" uri="urn:oasis:...:CommonAggregateComponents-2"/>
05 <include href="order-constraints.sch"/>
06 </schema>
```

### When no pattern is explicitly engaged, all patterns are engaged

- from all embedded patterns and included patterns

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Sample master that includes trading partner business rules with the select code lists

- total-constraints.sch
- used to pull in code lists
- includes reference to other business rules

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <schema xmlns="http://purl.oclc.org/dsdl/schematron"
03       defaultPhase="only-phase">
04   <title>Business rules for maximum total value</title>
05   <ns prefix="cbc" uri="urn:oasis:...:CommonBasicComponents-2"/>
06   <ns prefix="cac" uri="urn:oasis:...:CommonAggregateComponents-2"/>
07   <phase id="only-phase">
08       <active pattern="code-list-rules"/>
09       <active pattern="total-limit"/>
10   </phase>
11   <include href="total-limit-constraint.sch"/>
12   <include href="order-constraints.sch"/>
13 </schema>
```

### Schematron provides for invocation-time specification of phase to run

- each phase points to the active patterns to be engaged
- phases are not needed if all patterns are to be engaged

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Sample trading partner business rules

- total-limit-constraints.sch
- expresses arbitrary constraints on the information

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <pattern xmlns="http://purl.oclc.org/dsdl/schematron" id="total-limit">
03   <rule context="cbc:ToBePaidAmount">
04     <assert test="< . &lt; 10000">Total amount '<value-of select="."/>'
05 cannot be $10,000 or more</assert>
06   </rule>
07 </pattern>
```

Example business rule limits the total to be "10,000" or less

- note there is no indication of currency, though that could be added as a co-constraint

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Transcript of test run of second pass validation using the created pattern in both scenarios

```
01 Precondition validation...
02
03 Validating partner-agreed constraints...
04   w3cschema ContextValueAssociation.xsd order-constraints.cva
05 Attempting validating, namespace-aware parse
06 Parse succeeded (0.521) with no errors and no warnings.
07
08 Validating code lists...
09   w3cschema genericcode.xsd CA_CountrySubentityCode.gc
10 Attempting validating, namespace-aware parse
11 Parse succeeded (0.270) with no errors and no warnings.
12   w3cschema genericcode.xsd US_CountrySubentityCode.gc
13 Attempting validating, namespace-aware parse
14 Parse succeeded (0.330) with no errors and no warnings.
15   w3cschema genericcode.xsd CAUS_CurrencyCode.gc
16 Attempting validating, namespace-aware parse
17 Parse succeeded (0.260) with no errors and no warnings.
18   w3cschema genericcode.xsd TaxIdentifier.gc
19 Attempting validating, namespace-aware parse
20 Parse succeeded (0.260) with no errors and no warnings.
21   w3cschema genericcode.xsd Additional_PaymentMeansCode.gc
22 Attempting validating, namespace-aware parse
23 Parse succeeded (0.270) with no errors and no warnings.
24
25 Preparing code list rules...
26
27 Translating partner-agreed constraints into Schematron rules...
28 xslt order-constraints.cva ..\utility\Crane-UBL-genericcode2Schematron.xsl
29                                     order-constraints.sch
```

Of note above:

- note that the first pass schema structure validation is not included in this test script or transcript
- without first pass schema structure validation it is possible to get a "false positive" when information items are not in their correct places as expected for the contexts
- the order-constraints.cva file is translated into the order-constraints.sch file

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Transcript (cont.)

```
01 Test 1 - standalone code list rules
02
03 Assembling rules into a Schematron schema...
04 xslt codes-only-constraints.sch ..\utility\iso_schematron_assembly.xsl
05
06                                     order-codes-only.sch
07
08 Translating Schematron into validation stylesheet...
09 xslt order-codes-only.sch ..\utility\Message-Schematron-terminator.xsl
10
11                                     order-codes-only.xsl
12
13 Document validation...
14
15 Testing order-test-good1.xml...
16 xslt order-test-good1.xml order-codes-only.xsl nul
17 "2>test-constraints.txt"
18 Result: 0
19
20
21 Testing order-test-good2.xml...
22 xslt order-test-good2.xml order-codes-only.xsl nul
23 "2>test-constraints.txt"
24 Result: 0
```

Of note above:

- the "codes-only-constraints.sch" Schematron schema is assembled into the "order-codes-only.sch" Schematron schema, which in turn is translated into the "order-codes-only.xsl" XSLT 1 stylesheet
- the two XML instances "order-test-good1.xml" and "order-test-good2.xml" both have no violations of the context/value associations, so each validation process returns a zero return code

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Transcript (cont.)

```
01 Testing order-test-bad1.xml...
02 xslt order-test-bad1.xml order-codes-only.xsl nul
03 "2>test-constraints.txt"
04 Result: 1
05 Value supplied 'UYU' is unacceptable for values identified
06 by 'currency' in the context '@currencyID':
07 /Order/cac:TaxTotal[1]/cbc:TaxAmount[1]/@currencyID
08 Processing terminated by xsl:message at line 143
```

### Sample value violation 1: a bad currency value

- note the error message is the built-in wording
  - coded in the preparation stylesheets
- the currency code "UYU" for "Uruguayan Peso" is adjacent to USD in the currency code list
- in this scenario, perhaps the author of the document missed the choice for US dollars by accidentally picking the peso indicator
- a non-zero return code indicates that there is a context/value association violation
- the portion of the example instance with the error is below

```
01 ...
02 <cac:TaxTotal>
03   <cbc:TaxAmount currencyID="UYU">15.00</cbc:TaxAmount>
04   <cbc:TaxEvidenceIndicator>false</cbc:TaxEvidenceIndicator>
05   <cac:TaxSubTotal>
06     <cbc:TaxableAmount currencyID="USD">100.00</cbc:TaxableAmount>
07     <cbc:TaxAmount currencyID="USD">15.00</cbc:TaxAmount>
08     <cac:TaxCategory>
09       <cbc:ID>Z</cbc:ID>
10       <cbc:Percent>15</cbc:Percent>
11       <cac:TaxScheme>
12         <cbc:ID>Provincial Tax</cbc:ID>
13       </cac:TaxScheme>
14     </cac:TaxCategory>
15   </cac:TaxSubTotal>
16 </cac:TaxTotal>
17 ...
```



## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



## Transcript (cont.)

```
01 Testing order-test-bad2.xml...
02   xslt order-test-bad2.xml order-codes-only.xsl nul
03 "2>test-constraints.txt"
04 Result: 1
05 Invalid province or state 'ON' for buyer "Elliot's Electronics":
06 /Order/cac:BuyerCustomerParty[1]/cac:Party[1]/cac:Address[1]/
07 cbc:CountrySubentityCode[1]
08 Processing terminated by xsl:message at line 143
```

## Sample value violation 2: unexpected version of province code list

- note the error message is custom wording supplied in the CVA file (page 116)
  - message includes cited content
- the country sub-entity code "ON" for "Ontario" is correct, but the instance states the value "ON" is from the list version "1"
- the code list against which coded values are being checked is version "2"
  - see page 60 for the genericcode file for provinces
- one cannot assume that the code for a particular version of a list has the same semantic for that code in a different version of the list
- the portion of the example instance with the error is below

```
01 <cac:BuyerCustomerParty>
02   ...
03   <cac:Party>
04     <cac:PartyName>
05       <cbc:Name>Elliot's Electronics</cbc:Name>
06     </cac:PartyName>
07     <cac:Address>
08       <cbc:StreetName>Yonge St</cbc:StreetName>
09       <cbc:BuildingNumber>100</cbc:BuildingNumber>
10       <cbc:CityName>Toronto</cbc:CityName>
11       <cbc:PostalZone>M0H-0M0</cbc:PostalZone>
12       <cbc:CountrySubentityCode
13         listVersionID="1">ON</cbc:CountrySubentityCode>
14     </cac:Address>
15     <cac:Contact>
16       <cbc:Name>George Tirebiter</cbc:Name>
17     </cac:Contact>
18   </cac:Party>
19   ...
```

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



## Transcript (cont.)

```
01 Testing order-test-bad3.xml...
02   xslt order-test-bad3.xml order-codes-only.xsl nul
03 "2>test-constraints.txt"
04 Result: 1
05 Invalid state 'ON' for seller "Elliot's Electronics":
06 /Order/cac:SellerSupplierParty[1]/cac:Party[1]/
07 cbc:Address[1]/cbc:CountrySubentityCode[1]
08 Processing terminated by xsl:message at line 143
```

## Sample value violation 3: bad country sub-entity code for seller

- note all error messages citing XPath addresses use the authored prefixes
  - in this instance the element Order is using the default namespace, so has no prefix
- the seller's country sub-entity code cannot be a Canadian province, even if the version of the code is correct according to the genericcode file, because the genericcode file is not referenced for this context
- cbc:ToBePaidAmount is in error missing the ".", thus it is too large
- the portion of the example instance with the error is below

```
01 ...
02 <cac:SellerSupplierParty>
03   ...
04   <cac:Address>
05     <cbc:StreetName>Yonge St</cbc:StreetName>
06     <cbc:BuildingNumber>100</cbc:BuildingNumber>
07     <cbc:CityName>Toronto</cbc:CityName>
08     <cbc:PostalZone>M0H-0M0</cbc:PostalZone>
09     <cbc:CountrySubentityCode listVersionID="2">ON
10   </cbc:CountrySubentityCode>
11   </cac:Address>
12   ...
13 </cac:Party>
14 </cac:SellerSupplierParty>
15 ...
16 <cac:LegalTotal>
17   <cbc:LineExtensionAmount
18     currencyID="USD">100.00</cbc:LineExtensionAmount>
19   <cbc:TaxExclusiveAmount
20     currencyID="USD">100.00</cbc:TaxExclusiveAmount>
21   <cbc:TaxInclusiveAmount
22     currencyID="USD">115.00</cbc:TaxInclusiveAmount>
23   <cbc:ToBePaidAmount
24     currencyID="USD">11500</cbc:ToBePaidAmount>
25 </cac:LegalTotal>
```

## Code list methodology example (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Transcript (cont.)

```

01 Test 2 - with business rules
02
03 Assembling rules into a Schematron schema...
04   xslt total-constraints.sch ..\utility\iso_schematron_assembly.xsl
05                                     order-codes-total.sch
06
07 Translating Schematron into validation stylesheet...
08   xslt order-codes-total.sch ..\utility\Message-Schematron-terminator.xsl
09                                     order-codes-total.xsl
10
11 Document validation...
12
13 Testing order-test-bad3.xml...
14   xslt order-test-bad3.xml order-codes-total.xsl nul
15 "2>test-constraints.txt"
16 Result: 1
17 Total amount '11500' cannot be $10,000 or more:
18 /Order/cac:LegalTotal[1]/cbc:ToBePaidAmount[1]
19 Invalid state 'ON' for seller "Elliot's Electronics": /Order/
20 cac:SellerSupplierParty[1]/cac:Party[1]/cac:Address[1]/
21 cbc:CountrySubentityCode[1]
22
23 Processing terminated by xsl:message at line 144

```

Of note above:

- the "total-constraints.sch" Schematron schema is assembled into the "order-codes-total.sch" Schematron schema, which in turn is translated into the "order-codes-total.xsl" XSLT 1 stylesheet

Sample value violations 4: bad country sub-entity code for seller and bad business rule for total

- the XML instance tested has two violations, one a value list violation and the other a business rule violation, and the stylesheet returns a non-zero return code to indicate there is a problem

## Example combination of restriction and extension

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



Consider the scenario of needing to extend a restricted subset of UBL transportation status codes

- UBL lists 329 different transportation codes from UN/ECE Recommendation 24
- typically only a subset of standardized values are needed
  - requires the UBL code list to be subset into a custom restricted list
- in this scenario some non-standardized values are needed
  - requires the additional codes to be put into an extension list

For interoperability, whenever a UBL standardized code is used, UBL meta data is used

- this ensures that all applications geared for standardized UBL support will support a value specified with the proper UBL meta data

For proper identification additional codes should have non-UBL meta data

- this ensures there is no ambiguity between a future version of the UBL list and an additional code specified by trading partners
- if the value doesn't specify meta data then the application would probably interpret the value as being a UBL-standardized value

Scenario files found in the samp/transport/ subdirectory

## Example combination of restriction and extension (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



Step 1: work from original complete list in UBL-TransportationStatusCode-2.0.gc

- obtain the standardized UBL TransportationStatusCode genericcode file containing the complete suite of status codes

```

01 <gc:CodeList
xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
02 <Identification>
03 <ShortName>TransportationStatusCode</ShortName>
04 <LongName xml:lang="en">Transportation Status</LongName>
05 <LongName Identifier="listID">UN/ECE rec 24</LongName>
06 <Version>Third Revision</Version>
07 <CanonicalUri>urn:oasis:names:specification:ubl:codelist:gc:
TransportationStatusCode</CanonicalUri>
09 <CanonicalVersionUri>urn:oasis:names:specification:ubl:codelist:gc:
TransportationStatusCode-2.0</CanonicalVersionUri>
11 <LocationUri>http://docs.oasis-open.org/ubl/os-ubl-2.0/cl/gc/
12 default/TransportationStatusCode-2.0.gc</LocationUri>
13 <Agency>
14 <LongName xml:lang="en">United Nations Economic Commission for
15 Europe</LongName>
16 <Identifier>6</Identifier>
17 </Agency>
18 </Identification>
19 ...
20 <SimpleCodeList>
21 <Row>
22 <Value ColumnRef="code">
23 <SimpleValue>1</SimpleValue>
24 </Value>
25 <Value ColumnRef="name">
26 <SimpleValue>Arrival, completed</SimpleValue>
27 </Value>
28 </Row>
29 <Row>
30 <Value ColumnRef="code">
31 <SimpleValue>2</SimpleValue>
32 </Value>
33 <Value ColumnRef="name">
34 <SimpleValue>Loading, authorized</SimpleValue>
35 </Value>
36 </Row>
37 ...
38 </SimpleCodeList>
39 </gc:CodeList>

```

## Example combination of restriction and extension (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



Step 2: create reduced list My\_TransportationStatusCodes.gc

- create a reduced list with only the standardized codes desired from the full list
- use meta data that is different from the standardized meta data because this new list is, in fact, not the original complete list, and so should not be identified as the complete list

```

01 <gc:CodeList
xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
02 <Identification>
03 <ShortName>My_TransportationStatusCode</ShortName>
04 <LongName xml:lang="en">Abbreviated Transportation Status</LongName>
05 <Version>1</Version>
06 <CanonicalUri>urn:x-company:codes:gc:status</CanonicalUri>
07 <CanonicalVersionUri>urn:x-company:codes:gc:status-1
08 </CanonicalVersionUri>
09 </Identification>
10 ...
11 <SimpleCodeList>
12 <Row>
13 <Value ColumnRef="code">
14 <SimpleValue>1</SimpleValue>
15 </Value>
16 <Value ColumnRef="name">
17 <SimpleValue>Arrival, completed</SimpleValue>
18 </Value>
19 </Row>
20 <Row>
21 <Value ColumnRef="code">
22 <SimpleValue>5</SimpleValue>
23 </Value>
24 <Value ColumnRef="name">
25 <SimpleValue>Process, begun</SimpleValue>
26 </Value>
27 </Row>
28 <Row>
29 <Value ColumnRef="code">
30 <SimpleValue>49</SimpleValue>
31 </Value>
32 <Value ColumnRef="name">
33 <SimpleValue>Lost</SimpleValue>
34 </Value>
35 </Row>
36 </SimpleCodeList>
37 </gc:CodeList>

```

## Example combination of restriction and extension (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Step 3: create extension to values as standalone list

Additional\_TransportationStatusCodes.gc

- create a list of the additional codes that are not standardized
- use meta data that identifies this new list

```
01 <gc:CodeList
02   xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericcode/1.0/">
03   <Identification>
04     <ShortName>Additional_TransportationStatusCode</ShortName>
05     <LongName xml:lang="en">Additional Transportation Status</LongName>
06     <Version>1</Version>
07     <CanonicalUri>urn:x-company:codes:gc:addstatus</CanonicalUri>
08     <CanonicalVersionUri>urn:x-company:codes:gc:addstatus-1
09   </Identification>
10   ...
11   <SimpleCodeList>
12     <Row>
13       <Value ColumnRef="code">
14         <SimpleValue>998</SimpleValue>
15       </Value>
16       <Value ColumnRef="name">
17         <SimpleValue>Evaporated</SimpleValue>
18       </Value>
19     </Row>
20     <Row>
21       <Value ColumnRef="code">
22         <SimpleValue>999</SimpleValue>
23       </Value>
24       <Value ColumnRef="name">
25         <SimpleValue>Eaten</SimpleValue>
26       </Value>
27     </Row>
28   </SimpleCodeList>
29 </gc:CodeList>
```

## Example combination of restriction and extension (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Step 4: validate with reduced list and extension list status-constraints.cva

- declare the two lists using <ValueList> elements
  - union the two lists using the <Context> element
- the restricted list needs to masquerade the restricted list's meta data with the UBL list's meta data
  - this allows an instance to be validated with either the full UBL list or the reduced list, but any values not in the reduced list will be rejected during validation

```
01 <ValueListConstraints xmlns="http://docs.oasis-open.org/codelist/..."
02   xmlns:cbc="urn:oasis:...:CommonBasicComponents-2"
03   id="urn:x-exercise"
04   name="code-list-rules">
05
06   ... Annotation/Description/Title information ...
07
08   <!--list all of the genericcode expressions of agreed-upon
09     code list value enumerations-->
10   <ValueLists>
11     <ValueList xml:id="UBLcodes-restricted"
12       uri="My_TransportationStatusCodes.gc">
13       <!--masquerade the UBL meta data while pointing to reduced list-->
14       <Identification>
15         <ShortName>TransportationStatusCode</ShortName>
16         <LongName>Transportation Status</LongName>
17         <LongName Identifier="listID">UN/ECE rec 24</LongName>
18         <Version>Third Revision</Version>
19         <CanonicalUri>urn:oasis:names:specification:ubl:codelist:gc:
20 TransportationStatusCode</CanonicalUri>
21         <CanonicalVersionUri>urn:oasis:names:specification:ubl:codelist:gc:
22 TransportationStatusCode-2.0</CanonicalVersionUri>
23       <Agency>
24         <Name>United Nations Economic Commission for Europe</Name>
25         <AgencyID>6</AgencyID>
26       </Agency>
27     </Identification>
28   </ValueList>
29   <ValueList xml:id="extra-codes"
30     uri="Additional_TransportationStatusCodes.gc"/>
31 </ValueLists>
32 <!--list all of the contexts in which value enumerations are used-->
33 <Contexts>
34   <Context item="cbc:ConditionCode"
35     values="UBLcodes-restricted extra-codes"/>
36 </Contexts>
37 </ValueListConstraints>
```

## Example combination of restriction and extension (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



Step 5: create master schema `status-codes.sch` to incorporate list constraints

- create a schema fragment to include the constraints expressed by the context/value associations
- this modularity would allow other Schematron-expressed constraints to be included into the including fragment

```
01 <schema xmlns="http://purl.oclc.org/dsdl/schematron">
02   <title>Code list value assertions</title>
03   <ns prefix="cbc" uri="urn:oasis:...:CommonBasicComponents-2"/>
04
05   <include href="status-constraints.sch" />
06 </schema>
```

## Example combination of restriction and extension (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



Step 6:

- each section below is marked with "======" at the right edge of the transcript

Validate the constraints against the CVA document model

- this ensures that the expression of document context constraints is syntactically correct

Convert the constraints into a Schematron component

- this interprets the XPath addresses of the document contexts and genericcode expressions of codes and meta data
- masquerading meta data overrides any external genericcode meta data

Assemble Schematron components into a complete schema

- this creates a single Schematron schema from the including fragment and all included fragments

Convert Schematron schema into XSLT

- using the CVA wrapper and the publicly-available Schematron skeleton, this creates an XSLT stylesheet suitable for running against XML instances for value validation

```
01 :: Validate constraints against the CVAUG document model:=====
02 :: w3cschema ContextValueAssociation.xsd status-constraints.cva
03 :: Return: 0
04 Attempting validating, namespace-aware parse
05 Parse succeeded (0.270) with no errors and no warnings.
06 :: Convert constraints into a Schematron component:=====
07 :: xslt status-constraints.cva Crane-UBL-genericcode2Schematron.xsl
08 status-constraints.sch
09 :: Return: 0
10 :: Assemble Schematron components into a complete schema:=====
11 :: xslt status-codes.sch iso_schematron_assembly.xsl
12 status-codes-runtime.sch
13 :: Return: 0
14 :: Convert Schematron schema into XSLT:=====
15 :: xslt status-codes-runtime.sch Message-Schematron-terminator.xsl
16 status-codes-runtime.xsl
17 :: Return: 0
```

## Example combination of restriction and extension (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 2 - Code list methodology example



### Step 7:

- each section below is marked with "======" at the right edge of the transcript

Validate and check value constraints on message-good1.xml

- this instance validly uses a code from the standard UBL code list
- `<cbc:ConditionCode listID="UN/ECE rec 24">5</cbc:ConditionCode>`

Validate and check value constraints on message-good2.xml

- this instance validly uses a code from the extended code list with additional values
- note that while no meta data is given here, one could choose to add meta data provided it is the meta data of the extended list
- `<cbc:ConditionCode>999</cbc:ConditionCode>`

Validate and check value constraints on message-bad.xml

- this instance improperly uses a code from the extended code list with additional values while using meta data from the standard UBL code list
- `<cbc:ConditionCode listID="UN/ECE rec 24">999</cbc:ConditionCode>`

```
01 :: Validate and check constraints on message-good1.xml:=====
02 :: w3cschema UBL-TransportationStatus-2.0.xsd message-good1.xml
03 :: Return: 0
04 Attempting validating, namespace-aware parse
05 Parse succeeded (0.911) with no errors and no warnings.
06 :: xslt message-good1.xml status-codes-runtime.xsl nul
07 :: Return: 0
08 :: Validate and check constraints on message-good2.xml:=====
09 :: w3cschema UBL-TransportationStatus-2.0.xsd message-good2.xml
10 :: Return: 0
11 Attempting validating, namespace-aware parse
12 Parse succeeded (0.891) with no errors and no warnings.
13 :: xslt message-good2.xml status-codes-runtime.xsl nul
14 :: Return: 0
15 :: Validate and check constraints on message-bad.xml:=====
16 :: w3cschema UBL-TransportationStatus-2.0.xsd message-bad.xml
17 :: Return: 0
18 Attempting validating, namespace-aware parse
19 Parse succeeded (0.891) with no errors and no warnings.
20 :: xslt message-bad.xml status-codes-runtime.xsl nul
21 Value supplied '999' is unacceptable for values identified by
22 'UBLcodes-restricted extra-codes' in the context 'cbc:ConditionCode':
23 /TransportationStatus/cac:TransportEvent[2]/cac:CurrentStatus[1]/
24 cbc:ConditionCode[1]
25
26 Processing terminated by xsl:message at line 143
27 :: Return: 1
```

## Rendering context/value association files

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 3 - Rendering context/value association files



Some audiences do not appreciate having to read raw XML to interpret the contents

- angle brackets are distracting
- the volume of XML markup overwhelms the information content of the instance

Crane Softwrights Ltd. has published free developer resources with which to render a context/value association file to HTML

- XSLT 1.0 stylesheet: Crane-assoc2html.xsl
- standalone production of HTML from context/value association file
- browser-based viewing of HTML from context/value association file

The rendering of the context/value association file includes the rendering of the referenced genericcode files

- the stylesheet for the context/value association file automatically imports the stylesheet for a genericcode file

See Browser-based viewing of an HTML rendering (page 69) for the technique of embedding a stylesheet association processing instruction

## Rendering context/value association files (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 3 - Rendering context/value association files



Instead of embedding simple text for documentation, one can use rich markup

- the Crane stylesheet supports the embedding of HTML
- note in the order-constraints-doc.xml example the declaration of the HTML namespace and use of the "x:" namespace prefix

```

01 <?xml-stylesheet type="text/xsl" href="Crane-assoc2html.xsl"?>
02 <ValueListConstraints
03   xmlns="urn:oasis:names:tc:ubl:schema:Value-List-Constraints-0.8"
04   xmlns:cbc="urn:oasis:...:CommonBasicComponents-2"
05   xmlns:cac="urn:oasis:...:CommonAggregateComponents-2"
06   xmlns:x="http://www.w3.org/TR/REC-html40"
07   xmlns:sch="http://purl.oclc.org/dsdl/schematron"
08   id="urn:x-illustration"
09   name="code-list-rules">
10   <Title>
11     Illustration of code list constraints -
12     <x:samp>order-constraints.cva</x:samp>
13   </Title>
14   <Identification>
15     <x:pre>
16       $Id: order-constraints-doc.cva,v 1.1 2007/02/10 02:24:18
17       G. Ken Holman Exp $
18     </x:pre>
19   </Identification>
20   <Description>
21     <x:p>
22       This is an illustrative example of all of the features of
23       specifying the context/value constraints that one can express
24       for XML documents.
25     </x:p>
26     <x:p>
27       The validation requirements for this contrived scenario are as
28       follows:
29     <x:ul>
30       <x:li>the UN/CEFACT currency code list is restricted to be
31       only Canadian and US dollars:</x:li>
32       <x:li>the seller must be in the US</x:li>
33       <x:li>the buyer may be in either Canada or the US</x:li>
34       <x:li>the definition for Payment Means is extended to include
35       both UBL definitions and additional definitions</x:li>
36     </x:ul>
37   </x:p>
38 </Description>
39   ...

```

## Standalone production of an HTML rendering

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 3 - Rendering context/value association files



Consider the HTML rendering of the earlier example of a context/value association file on page 136 rendered using:

- java -jar saxon.jar -o order-constraints.html  
order-constraints.cva Crane-assoc2html.xsl

The resulting HTML file when rendered appears as follows:

### Illustration of code list constraints - order-constrai

name="code-list-rules" id="urn:x-illustration"

\$Id: order-constraints.cva,v 1.2 2007/02/08 04:15:26 G. Ken Holman Exp \$

This is an illustrative example of all of the features of specifying the context/value constraints that one can express for XML documents. The validation requirements for this contrived scenario are as follows: - the UN/CEFACT currency code list is restricted to be only Canadian and US dollars - the seller must be in the US - the buyer may be in either Canada or the US - the definition for Payment Means is extended to include both UBL definitions and additional definitions

#### Value Lists

xml:id="currency" uri="CAUS\_CurrencyCode.qc"

Restricted to only Canadian and US dollars.

Reference = CurrencyCode

Name = Currency

ID = ISO 4217 Alpha

URI = urn:un:unece:uncefact:odelist:specification:54217

Version = 2001

VersionURI = urn:un:unece:uncefact:odelist:specification:54217:2001

AgencyName = United Nations Economic Commission for Europe

AgencyID = 6

Note how the HTML browser renders the text together in a single paragraph

- there are no markup constructs in the documentary content
- there are line breaks in the resulting HTML page, but the browser normalizes these into a space, thus running the lines together



## Standalone production of an HTML rendering (cont.)

Chapter 5 - Associating controlled vocabularies in XML documents  
Section 3 - Rendering context/value association files



Compare with the rendering of the example of a context/value association file with embedded HTML on page 136:

### Illustration of code list constraints - order-constrai

`name="code-list-rules" id="urn:x-illustration"`

`$Id: order-constraints-doc.cva,v 1.1 2007/02/10 02:24:18 G. Ken Holma`

This is an illustrative example of all of the features of specifying the context/value constraints can express for XML documents.

The validation requirements for this contrived scenario are as follows:

- the UN/CEFACT currency code list is restricted to be only Canadian and US dollars
- the seller must be in the US
- the buyer may be in either Canada or the US
- the definition for Payment Means is extended to include both UBL definitions and definitions

#### Value Lists

`xml:id="currency" uri="CAUS.CurrencyCode.qc"`

```
Reference = CurrencyCode
Name      = Currency
ID        = ISO 4217 Alpha
URI       = urn:un:unece:uncefact:odelist:specification:54217
Version   = 2001
VersionURI = urn:un:unece:uncefact:odelist:specification:54217:2001
AgencyName = United Nations Economic Commission for Europe
AgencyID   = 6
```

Restricted to only Canadian and US dollars

Note how the HTML browser renders the embedded HTML constructs.

## Chapter 6 - Controlled vocabulary association detail



- Introduction - Controlled vocabulary association detail
- Section 1 - Context/value association files
- Section 2 - Masquerading meta data
- Section 3 - ISO/IEC 19757-3 Schematron
- Section 4 - Engaging value validation using Schematron

### Outcomes

- overview the XML document assertion schema language Schematron



## Controlled vocabulary association detail

Introduction - Chapter 6 - Controlled vocabulary association detail



CVA files are validated using an W3C Schema expression of constraints

- /xsd/ContextValueAssociation.xsd
- includes /xsd/xml.xsd

Development work still in progress

- the work is continuing by the OASIS technical committee
- needs more community input and practice before becoming version 1.0

## Context/value association vocabulary

Chapter 6 - Controlled vocabulary association detail

Section 1 - Context/value association files



<ValueListConstraints>

- document element
- temporary namespace
  - "http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/cd2-1.0/"
    - used only during committee draft specification development
    - when finalized will probably end with "/1.0/"
- name=
  - (mandatory name token) used to name the Schematron pattern created for these associations
- optional <Title>, <Identification>, <Description> and <Include> children
- mandatory <ValueLists> and <Contexts> children
- id=
  - (optional URI) used to give this resource a unique identifier
- version=
  - (optional) documentary information indicating the version of this resource
- queryBinding=
  - (optional) an indication of the addressing language used for document contexts
  - when not specified, it is up to the implementation to assume what the document context addressing language is

<Annotation>

- (optional) used to annotate the resource as a whole
- mixed content with foreign vocabularies is allowed to be used for both children
- <AppInfo>
  - (optional) an annotation targeted for downstream processing by an application
- <Description>
  - (optional) an annotation targeted for downstream reading by a human

<Title>

- (optional) value used in reports for titling information regarding the associations
- mixed content with foreign vocabularies is allowed to be used

## Context/value association vocabulary (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 1 - Context/value association files



## &lt;Include&gt;

- (optional and repeatable) directive used to include rules from another value list context association file
  - important for the management of many context and value associations
  - the including CVA file constraints have priority over included CVA file constraints for the same document context
- uri=
  - (mandatory) location of the file
- <Annotation>
  - (optional) annotations related to this directive

## Context/value association vocabulary (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 1 - Context/value association files



## &lt;ValueLists&gt;

- (mandatory) collection of pointers to external representations of value lists
- <Annotation>
  - (optional) annotations related to this construct

## &lt;ValueList&gt;

- (optional and repeatable) pointer to external representation of value lists
- xml:id=
  - (mandatory) unique identifier for this pointer to a resource
- uri=
  - (mandatory) external URI for this resource
- key=
  - (optional) identification of the key identifier in the genericcode file for code lookup
- <Annotation>
  - (optional) annotations related to this construct
- optional <Identification> child (at most one)
- mixed content with foreign vocabularies is allowed to be used for documentation

## &lt;Identification&gt;

- (optional) this child of <ValueList> defines masquerading meta data that overrides corresponding meta data found in the external file with the following optional child elements:
  - <Annotation>
    - (optional) annotations related to this construct
  - <ShortName>
  - <LongName>
  - <Version>
  - <CanonicalUri>
  - <CanonicalVersionUri>
  - <LocationUri>
  - <AlternateFormatLocationUri>
  - <Agency><ShortName>
  - <Agency><LongName>
  - <Agency><Identifier>
- the <Identification> element uses the identical element structure (with names in the context/value association namespace rather than no namespace) such that one can copy and paste the <Identification> element from a genericcode file
  - note per XML rules, when the context/value association namespace is assigned to a prefix, each of the elements copied will need to be prefixed

## Context/value association vocabulary (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 1 - Context/value association files



## &lt;Contexts&gt;

- (mandatory) collection of information item contexts
- <Annotation>
  - (optional) annotations related to this construct
- optional <Context> children

## &lt;Context&gt;

- a single information item context
- values=
  - (mandatory) space-separated pointer to unique identifiers of external value list representations associated with the information item in context
- item=
  - (mandatory) XPath location step for information item
  - either "ElementName" or "@attributeName" identifying the information item without any context
- scope=
  - (optional; mutually exclusive with address=) ancestral context for the item
  - does not make reference to the information item itself, only the ancestor of the information item
- address=
  - (optional; mutually exclusive with scope=) specific XPath location path context for the information item
  - must make reference to the information item itself as the rightmost location step
  - this is used in circumstances where ancestral context is either insufficient or too encompassing
- mark=
  - (optional) a name token (no spaces) characterizing a violation of this context for downstream purposes
    - can be used, for example, to distinguish warnings from errors
    - no predefined semantics of the values being used
    - a downstream process can base decision making on the nature of the mark on each violation
- <Annotation>
  - (optional) annotations related to this construct
- optional <Message> child (at most one)
- mixed content with foreign vocabularies is allowed to be used for documentation

## Context/value association vocabulary (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 1 - Context/value association files



## &lt;Message&gt;

- (optional) a Schematron reporting message replacing the default generated message
  - when a custom-worded message is needed when reporting a constraint violation
  - typically this includes Schematron constructs such as <sch:value-of> with which a tailored message can retrieve neighboring information from the document context of the constraint violation
- this element is ignored if it has no child elements or no non-white-space text strings

## Using scope= or address=

Chapter 6 - Controlled vocabulary association detail  
Section 1 - Context/value association files



### System-wide context (all documents)

- don't use either `scope=` or `address=`
- the matching XPath address is simply the `item=` information item name
- example:
  - `<Context item="@currencyID" values="currency">`
  - the `currencyID=` attribute is checked for the same values across all documents

### Single document-wide context

- use `scope=` with the document element
  - does not include the specification of the focus item in the context expression
- example:
  - `<Context item="@currencyID" scope="/in:Invoice" values="currencySell"/>`
  - `<Context item="@currencyID" scope="/po:Order" values="currencyBuy"/>`
  - the `currencyID=` attribute is checked with different values for orders than for invoices

### Sub-document context

- use `scope=` with an element other than the document element
  - does not include the specification of the focus item in the context expression
- example:
  - `<Context item="cbc:CountrySubentityCode" values="provinces states" scope="cac:BuyerCustomerParty">...`
  - `<Context item="cbc:CountrySubentityCode" values="states" scope="cac:SellerSupplierParty">...`
  - the `<cbc:CountrySubentityCode>` element is checked with different values in different sub-trees in the documents

## Using scope= or address= (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 1 - Context/value association files



### Narrow sub-document context

- use `address=` when using `scope=` is ambiguous
  - requires the specification of the focus item in the XPath expression
- very rarely needed, but important when a narrow specification is needed to disambiguate a broad specification
- example in support of the UBL document fragment below:
  - `<Context item="cbc:ID" values="categoryIdentifiers" address="cac:TaxCategory/cbc:ID"/>`
  - `<Context item="cbc:ID" values="schemeIdentifiers" address="cac:TaxScheme/cbc:ID"/>`
  - the `<cbc:ID>` element is checked in each explicitly-specified context
  - the use of `scope="cac:TaxCategory"` is inappropriate because there are at least two `cbc:ID` values descendent from that element
- note that the XPath address must be complete and include the target information item (found in `item=`) as the right-most step of the address (used in `address=`)

Consider the following UBL document fragment:

- note how the different `<cbc:ID>` elements are in overlapping sub-trees of the XML document

```

01 ...
02 <cac:TaxTotal>
03   <cbc:TaxAmount currencyID="UYU">15.00</cbc:TaxAmount>
04   <cbc:TaxEvidenceIndicator>>false</cbc:TaxEvidenceIndicator>
05   <cac:TaxSubTotal>
06     <cbc:TaxableAmount currencyID="USD">100.00</cbc:TaxableAmount>
07     <cbc:TaxAmount currencyID="USD">15.00</cbc:TaxAmount>
08     <cac:TaxCategory>
09       <cbc:ID>Z</cbc:ID>
10       <cbc:Percent>15</cbc:Percent>
11       <cac:TaxScheme>
12         <cbc:ID>Provincial Tax</cbc:ID>
13       </cac:TaxScheme>
14     </cac:TaxCategory>
15   </cac:TaxSubTotal>
16 </cac:TaxTotal>
17 ...

```

## Value list specification, extension and restriction

Chapter 6 - Controlled vocabulary association detail  
Section 1 - Context/value association files



Recall the overview of a context/value association file contents on page 102

- value lists are declared pointing to external value list enumerations
- contexts associate an XPath document context with one or more value list declarations

Recall the depiction of extending and restricting lists on page 103

- specification: simply pointing to the declaration of A
- extension/union: pointing to the declaration of the base list B and the declaration of the extension D
- restriction: pointing to the declaration of the restricted list D' that has masquerading meta data for the complete list D

```
01 <ValueListConstraints id="..." name="..." version="...">
02   <AppInfo>...</AppInfo>
03   <Description>...</Description>
04   <Title>...</Title>
05   <Include uri="..." />
06
07   <ValueLists>
08     <ValueList xml:id="A"   uri="A.gc" />
09     <ValueList xml:id="B"   uri="B.gc" />
10     <ValueList xml:id="C"   uri="C.gc" />
11     <ValueList xml:id="DP"  uri="DP.gc">
12       <Identification>...D meta data...</Identification>
13     </ValueList>
14   </ValueLists>
15
16   <Contexts>
17     <Context item="..." values="A">..simple..</Context>
18     <Context item="..." values="B C">..extending/unioning..</Context>
19     <Context item="..." values="DP">..restricting..</Context>
20   </Contexts>
21 </ValueListConstraints>
```

## Masquerading meta data

Chapter 6 - Controlled vocabulary association detail  
Section 2 - Masquerading meta data



All different value lists should have unique list-level meta data identifying each list

- e.g. every code list published by UBL has unique meta data
  - the supplied "pass 2" value validation checks instance-level meta data values against the list-level meta data
- the list-level meta data identifies the list properties including the list's custodian
- e.g. the code list for currency values

The list-level meta data for the ISO currency code list is as follows:

```
01 <Identification>
02   <ShortName>CurrencyCode</ShortName>
03   <LongName>Currency</LongName>
04   <LongName Identifier='listID'>ISO 4217 Alpha</LongName>
05   <Version>2001</Version>
06   <CanonicalUri>urn:un:unece:uncefact:codelist:
07 specification:54217</CanonicalUri>
08   <CanonicalVersionUri>urn:un:unece:uncefact:codelist:
09 specification:54217:2001</CanonicalVersionUri>
10   <Agency>
11     <LongName>United Nations Economic Commission for
12 Europe</LongName>
13     <Identifier>6</Identifier>
14   </Agency>
15 </Identification>
```

## Masquerading meta data (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 2 - Masquerading meta data



When restricting a published list, the restricted list has different meta data than the published list

- e.g. limiting the currency codes to only Canadian and US dollars

An example of list-level meta data for a reduced code list CAUS\_CurrencyCode.gc is as follows:

```
01 <Identification>
02 <ShortName>CAUSCurrencyCode</ShortName>
03 <LongName>Canadian and US Currency Codes</LongName>
04 <Version>1</Version>
05 <CanonicalUri>urn:x-company:CAUS-currency</CanonicalUri>
06 <CanonicalVersionUri>urn:x-company:CAUS-currency:1</CanonicalVersionUri>
07 </Identification>
```

Users specifying an item from the published list will typically be using for instance-level meta data the list-level meta data of the published list

- this ensures the XML document passes the supplied "pass 2" value validation checks for the published standardized document type
- alternatively trading partners may choose to require the meta data of the derived lists, though this "breaks" value validation against the standardized document type
- e.g. specifying a currency value with meta data from the published list of currency codes
  - <cbc:TaxableAmount currencyCodeListVersionID="2001" currencyID="USD">100.00</cbc:TaxableAmount>
- the user is expecting to validate the document against the published code list constraints

## Masquerading meta data (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 2 - Masquerading meta data



Masquerading meta data ensures a restricted list appears the same to validation as the published list

- the declaration of the external list of values incorporates the masquerading meta data in the declaration
  - recall the vocabulary for the declaration of a value list shown on page 143
  - this doesn't disturb the meta data in the restricted external files
  - trading partners can choose to use the declaration with or the declaration without the masquerading meta data
- the following example of a declaration to a constrained list of currency codes is identical to the list-level meta data found in the original list
  - simplest approach is just to copy the entire <Identification> element from the original list
    - noting, of course, that the elements in the genericcode file are in no namespace, but in the context/value association file they are in a namespace
    - if the CVA file is using a prefix for the CVA namespace, the copied elements will need to have each of their names appropriately prefixed
    - if the CVA file is using the default namespace for the CVA namespace, no changes are needed to the syntax

A declaration pointing to the reduced list while masquerading as the ISO list is as follows:

```
01 <ValueList xml:id="currency" uri="CAUS_CurrencyCode.gc">
02 <Annotation>
03 <Description>
04 Restricted to only Canadian and US dollars.
05 </Description>
06 </Annotation>
07 <Identification>
08 <ShortName>CurrencyCode</ShortName>
09 <LongName>Currency</LongName>
10 <LongName Identifier='listID'>ISO 4217 Alpha</LongName>
11 <Version>2001</Version>
12 <CanonicalUri>urn:un:unece:unefact:codelist:
13 specification:54217</CanonicalUri>
14 <CanonicalVersionUri>urn:un:unece:unefact:codelist:
15 specification:54217:2001</CanonicalVersionUri>
16 <Agency>
17 <LongName>United Nations Economic Commission for
18 Europe</LongName>
19 <Identifier>6</Identifier>
20 </Agency>
21 </Identification>
22 </ValueList>
```

## ISO/IEC 19757-3 Schematron

Chapter 6 - Controlled vocabulary association detail  
Section 3 - ISO/IEC 19757-3 Schematron



### Assertion-based validation schema language

- one asserts things that must be true about an XML instance
  - violations are reported when the assertion is false
- one reports things that must be false about an XML instance
  - violations are reported when the report is true
- based on the use of XPath specification of document contexts
  - recall the discussion on document contexts on page 99

### Contrasted with grammar-based or type-based validation schema language

- grammar-based validation schema languages
  - e.g. XML Document Type Definitions (DTD)
  - e.g. ISO/IEC 19757-2 Regular Language for XML (RELAX-NG)
- type-based validation schema language
  - e.g. W3C Schema

### Part of the Document Schema Definition Languages (DSDL)

- custody of ISO/IEC JTC 1/SC 34/WG 1
- ISO/IEC 19757 international standard with multiple parts:
  - <http://www.dsd1.org>
  - Part 1 - overview
  - Part 2 - RELAX-NG - regular-grammar-based validation
  - Part 3 - Schematron - assertion-based validation
  - Part 4 - Namespace-based Validation Dispatching Language (NVDL)
  - Part 5 - Data Type Library Language (DTLL)
  - Part 6 - Path-based Integrity Constraints
  - Part 7 - Character Repertoire Description Language (CRDL)
  - Part 8 - Document Schema Renaming Language (DSRL)
  - Part 9 - Datatype- and Namespace-aware DTD
  - Part 10 - Validation Management

## ISO Schematron deployment

Chapter 6 - Controlled vocabulary association detail  
Section 3 - ISO/IEC 19757-3 Schematron

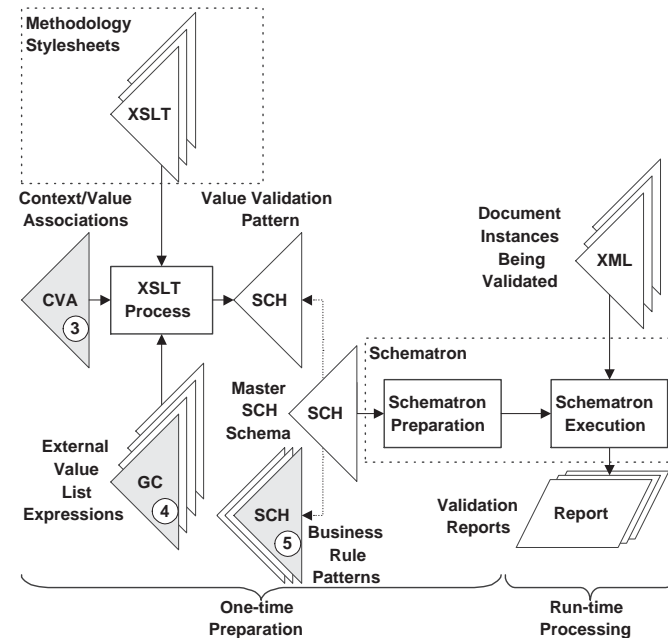


### Schematron schema preparation

- a master Schematron schema is created by the user to establish the Schematron environment for use of the included patterns
- fragmented Schematron schemas are assembled into a complete assertion
  - <sch:include> directive points to each schema fragment
  - the methodology stylesheets create a single Schematron pattern construct for inclusion for the contents of a context/value association file
  - other business rule Schematron schema fragments can be included
- the complete assertion schema is translated into a runtime artefact incorporating all assertion and report testing

### Schematron schema execution

- the runtime artefact is run against the XML instances to be tested, producing the validation reports
  - the nature of the report (i.e. text, XML, etc) is based on the choice of Schematron wrapper for the generated runtime artefact



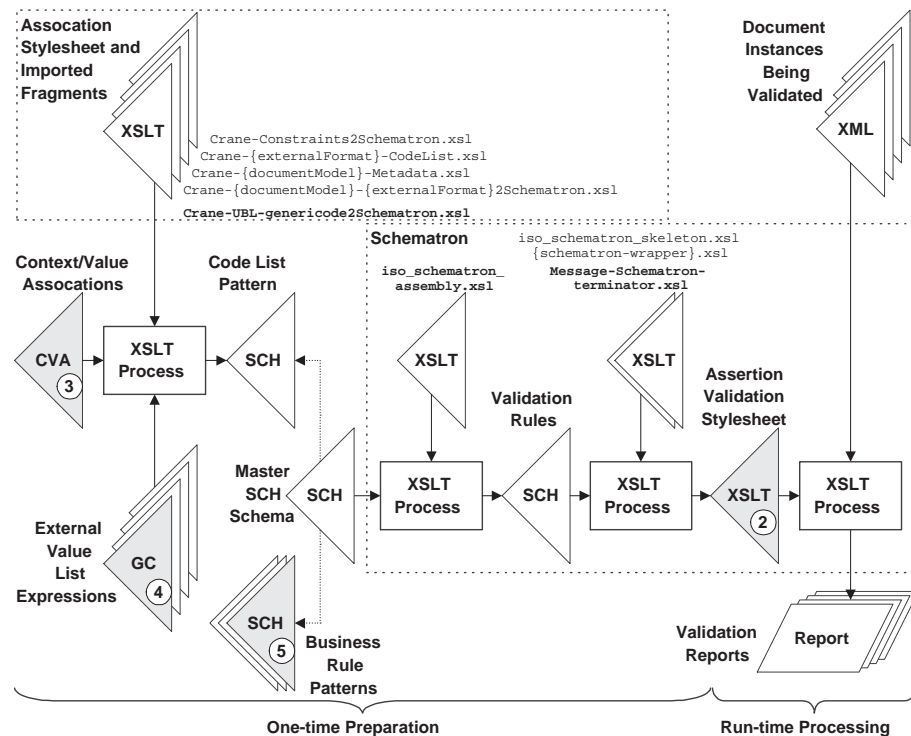
## ISO Schematron deployment (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 3 - ISO/IEC 19757-3 Schematron



An XSLT-based reference implementation of a Schematron skeleton is freely available

- <http://www.schematron.com>
- the skeleton implements base functionality
  - `iso_schematron_skeleton.xsl`
- the skeleton is wrapped by a stylesheet that tailors the reporting style
  - supplied with `Message-Schematron-terminator.xsl`
- instances being validated are processed with the resulting output stylesheet
  - validation results with a stylesheet generated using `Message-Schematron-terminator.xsl`:
    - text-based error report to the standard error port
    - testable non-zero return code on exit with error
    - does not exploit the `mark=` facility in CVA files



## ISO Schematron deployment (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 3 - ISO/IEC 19757-3 Schematron



Other Schematron wrappers can provide other reporting features

- **Schematron Validation Report Language (SVRL)**
  - an XML vocabulary for reporting Schematron validation results
  - stylesheets available for processing validation reports
  - a wrapper is already available for SVRL support
  - when `mark=` is used in the context/value association file, the Schematron implementation of this is the `flag=` attribute which is accessible when using SVRL
  - having the result of validation in an XML format enables one to process the results with XML-enabled tools
- one can create one's own wrappers
  - to meet custom requirements in a specific environment



## Schematron message construction

Chapter 6 - Controlled vocabulary association detail  
Section 3 - ISO/IEC 19757-3 Schematron



Not supplying a <Message> for a <Context> engages default messaging:

- the content in the following example is documentation, not a message

```
01 <Context item="@currencyID" values="currency">
02   <Annotation>
03     <Description>
04       All currencies are restricted to only Canadian and US dollars.
05     </Description>
06   </Annotation>
07 </Context>
```

The default message quotes the value in error and the values= specification of the value lists involved in which the value and associated meta data is not found:

```
01 Value supplied 'UYU' is unacceptable for values
02 identified by 'currency' in the context '@currencyID':
03 /Order/cac:TaxTotal[1]/cbc:TaxAmount[1]/@currencyID
```

The pro forma structure of the message is as follows:

```
01 Value supplied 'value-from-instance' is unacceptable for values
02 identified by 'id-of-genericcode-declaration' in the context
03 'document-context': instance-address-of-value
```

Note the lack of line numbers in the error message

- XML is not obliged to have line breaks and indents
- line numbers do not help when there are no line breaks in XML
- the XPath address is unambiguous
- some XML-based tools accept an XPath address as a lookup function

## Schematron message construction (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 3 - ISO/IEC 19757-3 Schematron



Using Schematron's <sch:value-of> one can craft a custom message:

- the <Message> element defines the content replacing the default message

```
01 <Context item="cbc:CountrySubentityCode"
02   scope="cac:SellerSupplierParty"
03   values="states">
04   <Annotation>
05     <Description>
06       The seller can only be in the US.
07     </Description>
08   </Annotation>
09   <Message>Invalid state '<sch:value-of select="."/>' for seller
10 "<sch:value-of select="ancestor::cac:SellerSupplierParty/
11 cac:Party/cac:PartyName/cbc:Name"/>"</Message>
12 </Context>
```

The specified message is used for the report, suffixed with the XPath expression:

```
01 Invalid state 'ON' for seller "Elliot's Electronics":
02 /Order/cac:SellerSupplierParty[1]/cac:Party[1]/cac:Address[1]/
03 cbc:CountrySubentityCode[1]
```

## Schematron message construction (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 3 - ISO/IEC 19757-3 Schematron



The inclusion of the path at the end of the message is an available feature for new wrappers

- the skeleton adds support for this to the validating stylesheet as an available mode for processing nodes
- the following has been proposed to be included in the skeleton for use by a wrapper stylesheet to engage the feature

```
01 <axsl:apply-templates select="." mode="schematron-get-full-path-3"/>
```

The above is only important when writing your own Schematron wrappers

- when using the Schematron implementation included with this methodology, there is no need to consider any changes in this regard

## Engaging value validation using Schematron

Chapter 6 - Controlled vocabulary association detail  
Section 4 - Engaging value validation using Schematron



Crane-UBL-genericcode2Schematron.xsl translates context/value associations for UBL into a Schematron pattern

- a pattern collects a group of rules
  - each rule constrains the value of a particular information item
- the pattern is given a name using the name= attribute in the context/value association file

Consider the example order-constraints.cva shown earlier:

```
01 <ValueListConstraints
02   xmlns="urn:oasis:names:tc:ubl:schema:Value-List-Constraints-0.8"
03   xmlns:cbc="urn:oasis:...:CommonBasicComponents-2"
04   xmlns:cac="urn:oasis:...:CommonAggregateComponents-2"
05   xmlns:sch="http://purl.oclc.org/dsdl/schematron"
06   id="urn:x-illustration"
07   name="code-list-rules">
08   ...
09 </ValueListConstraints>
```

This creates a Schematron pattern in order-constraints.sch as follows:

```
01 <pattern xmlns="http://purl.oclc.org/dsdl/schematron"
02         id="code-list-rules">
03   <!--
04   ...
05   <ns prefix="cbc" uri="urn:oasis:...:CommonBasicComponents-2"/>
06   <ns prefix="cac" uri="urn:oasis:...:CommonAggregateComponents-2"/>
07   -->
08   <rule context="@currencyID">
09     ...
10   </rule>
11 </pattern>
```

Note the embedded <ns> elements found in the comments

- these lines have no effect in the pattern because they are positioned inside of the comment
- these declarations are needed in the including master Schematron schema to satisfy the namespace requirements of the XPath addresses in the included patterns
- the prefixes are significant and should manually be harmonized across all included patterns for a single including master Schematron schema

Recall ISO Schematron deployment (page 154)

## Engaging value validation using Schematron (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 4 - Engaging value validation using Schematron



All Schematron patterns must be assembled into a single Schematron schema

- an including master Schematron schema points to all included patterns and other fragments
- the typical use is a single unnamed phase such that all patterns are engaged
- alternatively, the schema can describe multiple phases and selectively point to patterns in each phase
  - invocation of the validation stylesheet can then choose the utilized phase

Consider the example `codes-only-constraints.sch` shown earlier:

```
01 <schema xmlns="http://purl.oclc.org/dsdl/schematron">
02   <title>Code list value assertions</title>
03   <ns prefix="cbc" uri="urn:oasis:...:CommonBasicComponents-2"/>
04   <ns prefix="cac" uri="urn:oasis:...:CommonAggregateComponents-2"/>
05   <include href="order-constraints.sch"/>
06 </schema>
```

`iso_schematron_assembly.xsl` is used to interpret `<sch:include>`

- this produces `order-codes-only.sch` in the example

```
01 <schema xmlns="http://purl.oclc.org/dsdl/schematron">
02   <title>Code list value assertions</title>
03   <ns prefix="cbc" uri="urn:oasis:...:CommonBasicComponents-2"/>
04   <ns prefix="cac" uri="urn:oasis:...:CommonAggregateComponents-2"/>
05   <pattern id="code-list-rules">
06     ...
07     <rule context="@currencyID">
08       ...
09     </rule>
10   </pattern>
11 </schema>
```

Recall ISO Schematron deployment (page 154)

## Engaging value validation using Schematron (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 4 - Engaging value validation using Schematron



Business rules can be expressed in other patterns or in the including schema

- all of the patterns have equal weight
- the patterns can be grouped in phases
- matches specified in later patterns have higher priority than those in earlier patterns

Consider the example `total-constraints.sch` shown earlier:

```
01 <schema xmlns="http://purl.oclc.org/dsdl/schematron"
02   defaultPhase="only-phase">
03   <title>Business rules for maximum total value</title>
04   <ns prefix="cbc"
05     uri="urn:oasis:...:CommonBasicComponents-2"/>
06   <ns prefix="cac"
07     uri="urn:oasis:...:CommonAggregateComponents-2"/>
08   <phase id="only-phase">
09     <active pattern="code-list-rules"/>
10     <active pattern="total-limit"/>
11   </phase>
12   <include href="total-limit-constraint.sch"/>
13   <include href="order-constraints.sch"/>
14 </schema>
```

The `total-limit-constraints.sch` pattern has business rules

```
01 <pattern xmlns="http://purl.oclc.org/dsdl/schematron"
02   id="total-limit">
03   <rule context="cbc:ToBePaidAmount">
04     <assert test=". &lt; 10000">Total amount '<value-of select="."/>'
05 cannot be $10,000 or more</assert>
06   </rule>
07 </pattern>
```

`iso_schematron_assembly.xsl` is used to interpret `<sch:include>`

- this produces `order-codes-total.sch` in the example

## Engaging value validation using Schematron (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 4 - Engaging value validation using Schematron



An assembled Schematron schema is transformed into the runtime validation stylesheet

- Message-Schematron-terminator.xsl is the supplied transformation stylesheet
- the output of the transformation is the validation stylesheet acting on business document instances
- other wrappers for iso\_schematron\_skeleton.xsl are available from the Schematron web site

The order-codes-only.xsl is created for the example without business rules

- from order-codes-only.sch
  - created from codes-only-constraints.sch that includes order-constraints.sch
- checks only the constraints expressed in the context/value association file

The order-codes-total.xsl is created for the example with the business rule

- from order-codes-total.sch
  - created from total-constraints.sch that includes both order-constraints.sch and total-limit-constraints.sch
- checks the constraints expressed in the context/value association file and in the business rule

Recall the overview diagram in Context/value validation implementation (page 111)

## CVA hierarchy using Schematron

Chapter 6 - Controlled vocabulary association detail  
Section 4 - Engaging value validation using Schematron

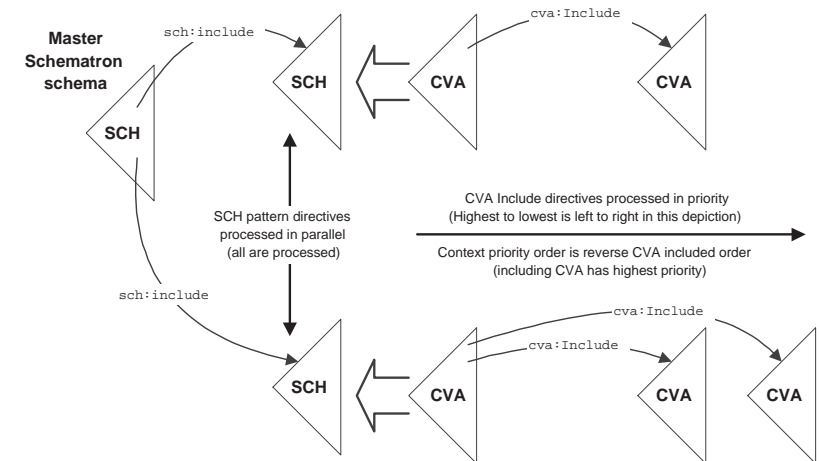


Each translated CVA file creates a Schematron pattern

- all included CVA files are incorporated into the one Schematron pattern
- the including association has priority over any included association
- for any given document context, only a single association will be tested
  - the first context in document order that matches

All included Schematron patterns are run in parallel

- actual order is irrelevant to validation, though the order of the error messages is impacted
- for all patterns, all document contexts will be tested, with only the highest priority association in each pattern for that context defining the test



## CVA hierarchy using Schematron (cont.)

Chapter 6 - Controlled vocabulary association detail  
Section 4 - Engaging value validation using Schematron



Deciding on the hierarchy depends on the desire for parallel or prioritized matching of document contexts

- use `cva:Include` for prioritized matching
  - the first match in document order for each tree of CVA files is tested
- use `sch:include` for parallel matching
  - highest priority match for each included Schematron schema is tested

Consider the situation where there are UBL, community and trading partner controlled vocabularies

- each group defines a CVA file for their controlled vocabularies
- use `sch:include` to run all tests simultaneously
- use `cva:Include` to run only the highest priority test
  - e.g. the trading partner CVA file would include the UBL CVA file and then the community CVA file if the prioritized order of matching is that the trading partner definitions have highest priority, followed by community definitions for the next priority, followed by the UBL definitions for the lowest priority

## Chapter 7 - Your own business document controlled vocabulary



- Introduction - Supporting your own business documents with context/value association files
- Section 1 - Adapting CVA2sch stylesheets for alternative environments

## Supporting your own business documents with context/value association files

Introduction - Chapter 7 - Your own business document controlled vocabulary



the CVA2sch validation stylesheets are delivered pre-configured for use with UBL

- genericode for the value list enumerations
- UBL conventions for the information item names
- UN/CEFACT CCTS conventions for the information item instance-level meta data

A "no metadata" configuration of the stylesheets is included with the methodology

- an off-the-shelf implementation of the methodology for business documents without information item meta data
- checks the information items identified in the CVA file without checking any instance-level meta data

Adaptation to other value list enumeration XML vocabularies

- a methodology stylesheet can be replaced with support for an arbitrary outboard representation of sets of enumerated values

Adaptation to other business document XML vocabulary instance-level meta data

- a methodology stylesheet can be replaced with support for arbitrary information item meta data

All methodology stylesheets are written in XSLT 1.0 for portability

- adaptation requires writing new stylesheet fragments to replace existing stylesheet fragments
- the existing stylesheet fragments are likely to be very useful as a model for the replacement
- XSLT 1 information item matching is only based on the namespace-qualified name
- XSLT 2.0 could be used for more nuanced item matching
  - schema-aware XSLT 2.0 could be used for type-based matching

## Adapting CVA2sch stylesheets for alternative environments

Chapter 7 - Your own business document controlled vocabulary

Section 1 - Adapting CVA2sch stylesheets for alternative environments



The stylesheet architecture is modular to allow a "plug and play" approach to using stylesheet fragments

- recall the overview diagram on page 111
- note the Association Stylesheet and Imported Fragments box at the top left

The naming the different components is based on a pattern for easy recognition

- Crane-{documentModel}-{externalFormat}2Schematron.xml
  - the wildcard stylesheet name for the fragment that imports the other stylesheet fragments for use with the a particular document vocabulary and a particular enumeration vocabulary
- Crane-UBL-genericode2Schematron.xml
  - the invocation stylesheet that imports the other stylesheet fragments for use with the UBL document vocabulary, UN/CEFACT CCTS instance-level meta data and the genericode enumeration vocabulary
- Crane-NM-genericode2Schematron.xml
  - the invocation stylesheet that imports the other stylesheet fragments for use with the an arbitrary document vocabulary without information item meta data and the genericode enumeration vocabulary
- Crane-{documentModel}-Metadata.xml
  - the wildcard stylesheet name for the imported fragment that supports the information item meta data for a particular document vocabulary
- Crane-UBL-Metadata.xml
  - the invoked stylesheet fragment for use with the UBL document vocabulary and UN/CEFACT CCTS instance-level meta data
- Crane-No-Metadata.xml
  - the invoked stylesheet fragment for use with the an arbitrary document vocabulary without information item meta data
- Crane-{externalFormat}-CodeList.xml
  - the wildcard stylesheet name for the imported fragment that supports a particular enumeration vocabulary
- Crane-genericode-CodeList.xml
  - the invoked stylesheet fragment for use with genericode expressions of enumerations
- Crane-Constraints2Schematron.xml
  - the invoked stylesheet fragment with common methodology functionality

## Adapting to alternative XML document formats

Chapter 7 - Your own business document controlled vocabulary  
Section 1 - Adapting CVA2sch stylesheets for alternative environments



The `Crane-{documentModel}-Metadata.xml` stylesheet needs to identify coded information items and interpret the instance-level meta data

- for each information item being tested, any anticipated attached instance-level meta data is passed to the common stylesheet fragment for processing

The document vocabulary for the XML instances should allow for the specification of information item meta data

- item meta data qualifies the item value as being from a particular controlled vocabulary as identified by the meta data values

The `Crane-No-Metadata.xml` stylesheet can be used when there is no instance-level meta data

- this fragment recognizes the instance-level meta data for information items of the source documents
  - inspects those information items governed by controlled vocabularies as indicated in the CVA file
  - specifies which instance-level meta data is associated with the information item
- without changes this module does not check any information items, thus all information items pass value validation purely on the coded value and not on any associated meta data

The `Crane-UBL-Metadata.xml` stylesheet identifies information items by their names

- assumes CCTS instance-level meta data according to the name (see page 48)

## Adapting to alternative XML enumeration formats

Chapter 7 - Your own business document controlled vocabulary  
Section 1 - Adapting CVA2sch stylesheets for alternative environments



The `Crane-genericcode-CodeList.xml` stylesheet is provided for genericcode support

- no changes are needed if using genericcode for the representation of code lists

Any `Crane-{externalFormat}-CodeList.xml` stylesheet can be written to interpret the external expression of list-level meta data

- creating such a module allows the methodology to be exploited for any XML representation of code lists other than genericcode

The context/value association file `<Identification>` element has a number of child elements to be mapped

- matching the genericcode identification components
  - can be adapted to other representations of list-level meta data
- other stylesheet fragments make the determination of the use of masquerading meta data
- the interface to this stylesheet fragment inquires the value of various parameters of meta data and returns each value on request

## Configuring the combination of stylesheet fragments

Chapter 7 - Your own business document controlled vocabulary  
Section 1 - Adapting CVA2sch stylesheets for alternative environments



The `Crane-{documentModel}-{externalFormat}2Schematron.xml` stylesheet specifies the fragments that are combined

- `Crane-UBL-genericcode2Schematron.xml`
  - the following fragments are included:
    - `Crane-UBL-Metadata.xml`
    - `Crane-genericcode-CodeList.xml`
    - `Crane-Constraints2Schematron.xml`
- `Crane-NM-genericcode2Schematron.xml`
  - the following fragments are included:
    - `Crane-No-Metadata.xml`
    - `Crane-genericcode-CodeList.xml`
    - `Crane-Constraints2Schematron.xml`

The `$comment` global variable adds documentary information

- the value of this variable is copied into the comments of the generated Schematron pattern file

## Annex A - OpenOffice 3 genericcode and CVA filters



- 
- Section 1 - OpenOffice 3 genericcode and CVA filters



## OpenOffice 3 genericode and CVA filters

Annex A - OpenOffice 3 genericode and CVA filters  
Section 1 - OpenOffice 3 genericode and CVA filters



Crane Softwrights Ltd.'s `gc2ods` package is a pair of OpenOffice 3 XML filters

- enables OpenOffice 3 to read and write genericode and context/value association files
- the user interface is presented in a spreadsheet document across multiple sheets

The package is found as a ZIP file linked from the sales page for this book

- <http://www.CraneSoftwrights.com/sales/pcli/>
- your book purchase password is needed to get access to the package

The `readme.html` documentation includes all of the necessary documentation to install, uninstall and use these filters.

## OpenOffice 3 genericode and CVA filters (cont.)

Annex A - OpenOffice 3 genericode and CVA filters  
Section 1 - OpenOffice 3 genericode and CVA filters



There are four tabs when editing genericode files:

- Identification
  - managing list-level meta data
- Columns
  - managing the kinds of value-level meta data
- Values
  - managing the actual codes and their associated value-level meta data values
- Help
  - context-sensitive help information

There are four tabs when editing context/value association files:

- Identification
  - managing association-wide meta data
- Values
  - managing the external value lists expressed in genericode
- Contexts
  - managing the document context items and their associated values
- Help
  - context-sensitive help information

## Where to go from here?

Conclusion - Practical Code List Implementation



The work on genericcode and context/value association continues:

- OASIS Genericcode 1.0 Committee Specification - December 27, 2007
  - <http://docs.oasis-open.org/codelist/genericcode>
- OASIS Context/value Association Specification - 0.5 draft 1
  - [http://www.oasis-open.org/committees/document.php?document\\_id=29990](http://www.oasis-open.org/committees/document.php?document_id=29990)
- focus now shifts to association, support, deployment, awareness and evangelism
- committee mail list - OASIS Code List Representation TC:
  - <http://lists.oasis-open.org/archives/codelist/>
- community mail list - CLR-Dev
  - <http://lists.oasis-open.org/archives/clr-dev/>
  - <http://www.oasis-open.org/mlmanage/>

## Colophon

Conclusion - Practical Code List Implementation



These materials were produced using structured information technologies as follows:

- authored source materials
  - content in numerous XML files maintained as external general entities for a complete prose book that can be made into a subset for training
    - specification of applicability of constructs for each configuration
      - 45- and 90-minute lecture, half-, full-, two- and three-day lecture and hands-on instruction, and book (prose) configurations
    - an XSLT transformation creates the subset of effective constructs from applying applicability to the complete file
    - content from other presentations/tutorials included semantically (not syntactically) during construct assembly
  - customized appearance engaged with marked sections and both parameter and general entities
    - different host company logos and venue and date marginalia
    - changing a single external parameter entity to a key file includes suite of files for given appearance
- accessible rendition in HTML
  - an XSLT stylesheet produces a collection of HTML files using Saxon for multiple file output
  - mono-spaced fonts and list-depth notation conventions assist the comprehension of the material when using screen-reader software
- printed handout deliverables
  - an XSLT stylesheet produces an instance of XSL formatting objects (XSL-FO) for rendering
  - XPDF <http://www.foolabs.com/xpdf> extracts raw text from PDF files for the back-of-the-book index methodology published as a free resource by Crane Softwrights Ltd.
  - XEP by RenderX <http://www.renderx.com> produces PostScript from XSL-FO
  - GhostScript <http://www.GhostScript.com> produces PDF from PostScript
  - the iText <http://itext.sf.net> PDF manipulation library for Java is used for page imposition by a custom Python <http://www.python.org> program running under the Jython <http://www.jython.org> environment

## Obtaining a copy of this material

Conclusion - Practical Code List Implementation



This comprehensive tutorial on code lists in XML is available for subscription purchase and free preview download:

- "Practical Code List Implementation" First Edition - 2009-02-09 - ISBN 978-1-894049-22-1
  - the free download preview excerpt of the publication includes the complete text of the first chapter and the introductory text of all of the other chapters
- the cost of purchase includes all future updates to the materials with email notification
  - the materials are updated after new content developed
    - more frequent in earlier editions than later editions
  - the materials are updated after incorporating comments gleaned during presentations and from feedback from customers
- available in PDF
  - formatted as 1-up or 2-up book pages per imaged page
  - dimensions in either US-letter or A4 page sizes
  - available as either single sided or double sided
- accessible rendition available for use with screen readers
- site-wide and world-wide staff licenses (one-time fee) are available

See <http://www.CraneSoftwrights.com/links/trn-20090209.htm> for more details.

Software available to customers

- accompanying software is available at no charge to customers of this book
- the license for this free software does not allow for free distribution of the software to others
- free download from <http://www.CraneSoftwrights.com/sales/pcli/> to registered users

Feedback

- the unorthodox style has been well-accepted by customers as an efficient learning presentation
- feedback from customers is important to improve or repair the content for future editions
- please send suggestions or comments (positive or negative) to [info@CraneSoftwrights.com](mailto:info@CraneSoftwrights.com)



## Practical Code List Implementation (Using Controlled Vocabularies in XML Documents)

Crane Softwrights Ltd.

<http://www.CraneSoftwrights.com>

## Table of contents

Indexed by slide number



1	[Prelude] Practical Code List Implementation (2) (3)
4	[Overview] Practical Code List Implementation
5	[Introduction I-1] Practical Code List Implementation
6	[1] Controlled vocabularies
7	[Introduction 1-I-1] XML document interchange
8	[Introduction 1-I-2] Controlled vocabulary semantics (9) (10)
11	[1-1-1] Codes and identifiers
12	[1-1-2] Code list registration authorities
13	[1-1-3] Identifying controlled vocabularies (14) (15) (16) (17)
18	[1-1-4] Modeling controlled vocabularies
19	[1-1-5] Expressing controlled vocabularies (20)
21	[1-1-6] Data entry of controlled vocabularies
22	[1-1-7] Application development supporting controlled vocabularies
23	[1-1-8] Validating controlled vocabularies (24) (25)
26	[1-1-9] Semantic representation by fixed values
27	[1-1-10] Trading partners and agreements (28)
29	[2] Defining and using controlled vocabularies
30	[Introduction 2-I-1] Controlled value list maintenance and identity
31	[Introduction 2-I-2] Controlled value specification
32	[2-1-1] Controlled XML information item value specification
33	[2-1-2] Document context
34	[2-1-3] Controlled value information item validation (35)
36	[2-1-4] Layered constraints on business documents
37	[2-2-1] Example: controlled vocabularies in UBL (38) (39)
40	[2-3-1] Declaration techniques
41	[2-3-2] UN/CEFACT controlled-vocabulary declarations
42	[2-3-3] UN/CEFACT instance-level meta data (43)
44	[2-3-4] Genericcode controlled-vocabulary declarations
45	[2-3-5] Examples and locations of UBL code list definitions
46	[2-3-6] Instance-level meta data in XML instances (47) (48) (49)
50	[2-3-7] Polaris controlled-vocabulary declarations (51) (52)
53	[3] Declaring controlled vocabularies
54	[Introduction 3-I-1] Declaring controlled vocabularies
55	[3-1-1] OASIS Genericcode 1.0 (56) (57) (58)
59	[3-1-2] Example code list validation scenario
60	[3-1-3] Specifying a controlled vocabulary (61)
62	[3-2-1] OASIS Context/value association using genericcode
63	[3-2-2] Extending a controlled vocabulary (64)
65	[3-2-3] Restricting a controlled vocabulary (66)
67	[3-3-1] Rendering controlled vocabularies
68	[3-3-2] Standalone production of an HTML rendering
69	[3-3-3] Browser-based viewing of an HTML rendering
70	[3-3-4] Problems triggering browser-based rendering (71)
72	[4] Controlled vocabulary representation detail
73	[Introduction 4-I-1] Controlled vocabulary representation detail

74	[4-1-1] Genericcode information
75	[4-1-2] Genericcode list-level meta data
76	[4-1-3] Genericcode standalone simple enumeration sets
77	[4-1-4] Genericcode XML (78) (79) (80) (81) (82) (83) (84) (85) (86)
87	[4-2-1] Mapping genericcode meta data to XML instances (88)
89	[5] Associating controlled vocabularies in XML documents
90	[Introduction 5-I-1] Constraining information items using controlled vocabularies
91	[Introduction 5-I-2] Context/value association (92) (93)
94	[Introduction 5-I-3] Using context/value association for validation (95) (96)
97	[5-1-1] CVA for validation using Schematron (98)
99	[5-1-2] Document context representation (100) (101)
102	[5-1-3] Context/value association specification
103	[5-1-4] Association specification, extension and restriction (104)
105	[5-1-5] Context/value pro forma associations (106) (107) (108) (109)
110	[5-1-6] Context/value validation implementation (111)
112	[5-2-1] Example code list validation scenarios
113	[5-2-2] Code list methodology example (114) (115) (116) (117) (118) (119) (120) (121) (122) (123) (124) (125) (126)
127	[5-2-3] Example combination of restriction and extension (128) (129) (130) (131) (132) (133) (134)
135	[5-3-1] Rendering context/value association files (136)
137	[5-3-2] Standalone production of an HTML rendering (138)
139	[6] Controlled vocabulary association detail
140	[Introduction 6-I-1] Controlled vocabulary association detail
141	[6-1-1] Context/value association vocabulary (142) (143) (144) (145)
146	[6-1-2] Using scope= or address= (147)
148	[6-1-3] Value list specification, extension and restriction
149	[6-2-1] Masquerading meta data (150) (151)
152	[6-3-1] ISO/IEC 19757-3 Schematron
153	[6-3-2] ISO Schematron deployment (154) (155)
156	[6-3-3] Schematron message construction (157) (158)
159	[6-4-1] Engaging value validation using Schematron (160) (161) (162)
163	[6-4-2] CVA hierarchy using Schematron (164)
165	[7] Your own business document controlled vocabulary
166	[Introduction 7-I-1] Supporting your own business documents with context/value association files
167	[7-1-1] Adapting CVA2sch stylesheets for alternative environments
168	[7-1-2] Adapting to alternative XML document formats
169	[7-1-3] Adapting to alternative XML enumeration formats
170	[7-1-4] Configuring the combination of stylesheet fragments
171	[A] OpenOffice 3 genericcode and CVA filters
172	[A-1-1] OpenOffice 3 genericcode and CVA filters (173)
174	[Conclusion C-1] Where to go from here?
175	[Conclusion C-2] Colophon
176	[Conclusion C-3] Obtaining a copy of this material
177	[Postlude] Practical Code List Implementation

## Index



**A**  
 account identifiers 28  
 agency 75  
 <Agency>  
   in genericcode files 80  
   in context/value association files 143  
 algorithmic 11  
 <AlternateFormatLocationUri>  
   in genericcode files 79  
   in context/value association files 143  
 <Annotation>  
   in genericcode files 82, 83, 84, 85  
   in context/value association files 102, 105, 141, 142, 143, 144  
 <AppInfo>  
   in genericcode files 83, 84, 85  
 application 8, 22, 35, 54, 94

**B**  
 business rules 25, 28, 34, 90, 96, 98, 112, 119, 120, 153, 161

**C**  
 <CanonicalUri>  
   in genericcode files 79, 82, 83  
   in context/value association files 143  
 <CanonicalVersionUri>  
   in genericcode files 79, 82, 83  
   in context/value association files 143  
 characteristic 11  
 code 9, 10, 29, 42, 76  
 code list 74, see also controlled vocabulary  
 code list set 74  
 <CodeList> 74, 77  
 <CodeListSet>  
   in genericcode files 74, 86  
 colloquial XML vocabulary 12  
 <Column> 82  
 column set 74, 76  
 <ColumnRef>  
   in genericcode files 83, 85  
 <ColumnSet>  
   in genericcode files 74, 81, 82, 83, 86  
 comma separated values (CSV) 12, 19, 30  
 community 16  
 <ComplexValue> 85

constraints 24  
   structural constraints 90  
   value constraints 90  
 container size code list 11, 12, 37, 38, 45, 55, 57  
 <Context> 102, 107, 144, 146, 156  
 context (document) 26, 33, 98, 99, 100, 101, 102, 108, 146, 147  
 <Contexts> 107, 141, 144  
 context/value association 25, 62, 63, 64, 65, 66, 74, 89, 139  
 controlled vocabulary 11  
 Core Component Technical Specification (CCTS) 32, 38, 40, 41, 42, 43, 45, 48, 166, 167, 168  
 country codes 8, 10, 13, 30, 39  
 Crane-gc2ods 3, 5, 171  
 currency codes 8, 10, 13, 26, 39, 41, 45, 48, 55, 56, 66, 98, 149  
 custodian 12

**D**  
 <Data>  
   in genericcode files 82  
 data entry 21, 93  
 database 9, 10, 12, 23, 30, 54  
 defaultCodeList.xsl 24, 39, 45, 94, 98  
 <Description>  
   in genericcode files 83, 84, 85  
   in context/value association files 102, 105, 141  
 Document Schema Definition Languages (DSDL) 152  
 document status codes 28, 30, 45  
 Document Type Definition (DTD) 23, 152

**E**  
 enumeration 55  
 extending code lists 63, 64, 103, 104, 130, 148  
 Extensible Stylesheet Language Transformations (XSLT) 54, 67, 95, 96, 97, 98, 110, 111, 166

**F**  
 file extension 70, 71

**G**  
 genericcode 20, 25, 54, 55, 56, 57, 58, 59, 60, 61, 72, 92, 169

**H**  
 hierarchies in an XML document 7  
 Hypertext Markup Language (HTML) 67, 68, 69, 97, 135

**I**  
 <Identification>  
   in genericcode files 77, 78, 79, 80  
   in context/value association files 141, 143  
 identifier 29, 43, 75  
 <Identifier>  
   in genericcode files 80  
 Identifier="listID" 78  
 <sch:include> 102, 153  
 <Include> 102, 105, 141, 142  
 interchange 19, 27  
 International Organization for Standardization (ISO) 8, 9  
 International Standard Book Numbering (ISBN) 12, 30  
 Internet Explorer (IE) 70

**J**

**K**  
 <Key> 83  
 key code 9, 18, 20, 31, 55, 74, 76  
 key= in context/value association files 81

**L**  
 language 10  
 legend 2  
 <LocationUri>  
   in genericcode files 79  
   in context/value association files 143  
 long name 75  
 <LongName>  
   in genericcode files 78, 80, 82, 83  
   in context/value association files 143  
 lookup value 11

**M**  
 masquerade 65, 91, 103, 104, 131, 149, 150, 151, 169  
 master Schematron schema 118, 132, 153  
 measurement identifiers 28  
 <Message> 144, 145, 156, 157, 158

meta data  
   instance-level 16, 17, 20, 21, 26, 42, 43, 46, 47, 48, 49, 50, 63, 65, 87, 93, 111, 166, 167, 168  
   list-level 9, 14, 17, 20, 26, 47, 55, 63, 64, 65, 66, 73, 75, 87, 88, 93, 149, 169  
   meta-data-only code list 54  
   omitting meta data 46  
   value-level 9, 10, 15, 17, 20, 21, 26, 31, 55, 73, 76, 93  
 Multimedia Internet Mail Extension 38, 39, 41

**N**  
 name 9, 55, 76  
 Naming and Design Rules (NDR) 40, 99  
 normalization of text 38

**O**  
 OpenOffice 3 171

**P**  
 <Parameter>  
   in genericcode files 82  
 payment means codes 8, 10, 13, 26, 28, 30, 39, 44, 45, 50, 64  
 Polaris 50, 51, 52  
 Python 95, 97

**Q**

**R**  
 RELAX-NG (Regular Language for XML) 23, 152  
 rendering 67, 68, 69, 70, 71, 135, 136, 137, 138  
 restricting code lists 65, 66, 103, 104, 129, 148  
 <Row> 85

**S**  
 sample code fragments 2  
 Schematron 23, 24, 25, 73, 91, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 145, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164  
   pattern 160, 163  
 Schematron Validation Report Language (SVRL) 155  
 scope 20  
   document-wide scope 20

- semantics (meaning) 10, 13, 18, 26, 27, 31
- short name 75
- <ShortName>
  - in genericcode files 78, 80, 82, 83
  - in context/value association files 143
- <SimpleCodeList> 77, 84, 85
- <SimpleValue> 85
- stylesheet
  - association 69
  - modularization 87, 167
- supplementary components 38, 48
- T**
  - <Title> 102, 105, 141
- trading partners 8, 27, 28, 34, 90, 97, 112
  - agreements/contracts 8, 11, 19
  - relationship 22, 35, 93
- translations 10, 18
- transportation status codes 28, 127, 128, 129, 130, 131, 132, 133, 134
- typographical conventions 2
- U**
  - UN/CEFACT 8, 32, 40, 41, 42, 43, 48, 88
- unconstrained code lists 13, 38
- union
  - of code lists 21, 31, 62, 63, 131
- uniqueness 9, 31
- unit of measure codes 8, 39, 41
- Universal Business Language (UBL) 11, 12, 24, 32, 36, 37, 38, 39, 55, 74, 88, 94, 97, 98, 101, 104, 111, 127, 128, 129, 130, 131, 132, 133, 134, 147, 159, 166, 167, 168
- user interface 21
- V**
  - validation 23, 24, 25, 34, 35, 59, 92, 93, 94, 95, 96, 159, 160, 161, 162, 163, 164
  - structural/lexical 34
  - value 34
  - <Value> 85
  - <sch:value-of> 157
  - <ValueList> 102, 106, 143
  - <ValueListConstraints> 141
  - <ValueLists> 106, 141, 143
  - <Version>
    - in genericcode files 79
    - in context/value association files 143
- versions of code lists 63, 75
- vocabulary, controlled 7
  - semantics 8, 9, 10
- vocabulary, XML 7
- W**
  - W3C Schema 8, 12, 20, 23, 33, 40, 41, 50, 58, 73, 90, 94, 97, 140, 152
- white-space characters 38
- X**
  - XML Path Language (XPath) 25, 92, 99, 100, 101, 105, 109, 147, 152
- Y**
- Z**